Techniques for Enhancing the Computational Speed of Multiple Object Tracking

by

Gurinderbeer Singh

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs in partial fulfillment of the requirements for the degree of

Master of Applied Science

in

Electrical and Computer Engineering

Carleton University
Ottawa, Ontario

Abstract

A massive amount of video data is recorded daily for forensic post analysis and computer vision applications that are used for the analysis of this video data, often perform Multiple Object Tracking (MOT). Advancements in image analysis algorithms and global optimization techniques have improved the accuracy of MOT, often at the cost of slow processing speed which limits its applications to small video datasets. With a focus on fast MOT, this thesis introduces a greedy data association technique (GDA) for MOT, which finds a locally optimum solution with a low computational overhead. To further enhance the computational speed of MOT for large video datasets, three MapReduce-based parallel techniques are introduced. The performance analysis using a set of benchmark video datasets with system prototypes shows that the proposed techniques are significantly faster than the existing state-of-the-art methods.

Acknowledgements

I would like to express my special thanks of gratitude to my supervisors Dr. Shikharesh Majumdar and Dr. Sreeraman Rajan for their immeasurable guidance, encouragement and financial support throughout the whole process of this thesis research. I would also like to thank my family for the continuous moral support throughout my master's degree. Finally, I would like to thank all my friends, especially Rudraneel Chakraborty and Navgeet Chhatwal, for every kind of help and support they provided during the completion of this thesis research.

Table of Contents

List of Figures

List of Tables

List of Abbreviations and Symbols

$A_{COST}(T_i,T_j)$	Association cost between the tracklets T_i and T_j
AWS	Amazon Web Services
CCTV	Closed-Circuit Television
CLI	Command Line Interface
CPU	Central Processing Unit
$d_i^{f_i}$	An object detection in frame f_i
D	A set of object detections
EC2	Elastic Compute Cloud
E_N	System Efficiency for N processors
f	A frame of a video sequence
FNs	Number of false negatives
FPs	Number of false positives
FP	Fully Parallel Technique
FPO	Fully Parallel Technique with Frame Overlap
GDA	A Greedy Data Association Technique for Multiple Object Tracking
G_{max}	Maximum temporal gap (frame gap)
GOP	Group of Pictures
GPU	Graphics Processing Unit
HD	High-Definition
HDFS	Hadoop Distributed File System
Hz	Processing speed in frames per second

IDS	Number of Identity Switches
IoT	Internet of Things
$L(d_i,d_j)$	A link cost between the detections d_i and d_j
MB	Megabytes
ML	Proportion of mostly lost targets
MOT	Multiple Object Tracking
MOT-MR	MapReduce-based Parallel Techniques for Multiple Object Tracking
MOT4	MOT16-04 sequence of MOT16 dataset
MOTA	Multiple Object Tracking Accuracy
MOTP	Multiple Object Tracking Precision
MT	Proportion of mostly tracked targets
$P_j^{f_i}$	An estimated position of Tracklet T_j at frame f_i
PETS	PETS2009-S2L1 sequence of PETS2009 dataset
PP	Partially Parallel Technique
R-CNN	Region-based Convolutional Neural Network
S	A set of object tracklets
S_N	Computational Speedup for N processors
T	An object trajectory/tracklet
TBD	Tracking-by-Detection
TCP	Transmission Control Protocol
V_j	Velocity of Tracklet T_j

YARN	Yet Another Resource Negotiator
YOLO	You Only Look Once
θ	A threshold value

Chapter: 1 Introduction

In the era of Internet of Things (IoT) and Big Data, a massive amount of video data gets recorded, for instance, in video surveillance, traffic management, and sports analysis. The video data needs to be analyzed for obtaining useful information. As a result, research in the field of video analytics has gained immense popularity, both in academics and industry. Multiple Object Tracking (MOT) is one of the several topics which comes under this field. It plays a major role in applications such as automated visual security and surveillance [1], traffic control [2], smart vehicles [3], robotics [4], and medical imaging [5].

There are two stages in MOT: the object detection stage and the data association stage. The former is a time-independent process of obtaining spatial locations of objects (object detections) present in each frame in a video sequence while latter is a time-dependent process of associating object detections obtained from the object detection stage into object trajectories that span multiple frames [6]. The object detection stage is a time-independent process because it does not have inter-frame dependencies, that is, is each frame of the video sequence is processed independently of other frames. The data association stage is a time-dependent process because it has inter-frame dependencies. The data association stage gets the object detections from the object detection stage and associates these detections across the frames by using a data association algorithm. The basic idea of the data association algorithm is to assign a same unique identity to all the detections that belong to one object and generate continuous object trajectories (tracklets) across the frames. Since the data association stage utilizes object motion and object appearances from across the

frames to associate the object detections, the data associations in each frame have dependencies on the adjacent frames.

There are two broad categories of MOT algorithms: real-time (online tracking) and batch processing (offline tracking). Online tracking deals with live stream videos. The data association stage in the online tracking uses only the current and past frames. Offline tracking incorporates global object detections from all the frames of a pre-recorded video sequence.

Video analytics such as MOT take a considerable amount of computational time. This thesis focuses on providing techniques that enhance the computational speed of MOT. The following section provides the motivation for developing such techniques.

1.1 Motivation for the Thesis

A number of state-of-the-art object detection algorithms [7, 8, 9] and data association algorithms [6, 10, 11, 12, 13] that provide a good qualitative performance of MOT exist in the literature. These algorithms focus on improving the qualitative performance metrics such as multiple object tracking accuracy (MOTA), number of false positives (FPs), number of false negatives (FNs) and identity switches (IDS) that are described in Chapter 3. To achieve higher qualitative performance, these algorithms utilize complex image analysis and/or global optimization techniques and thus have a high computational cost. For instance, the object detection method proposed in [14] can take up to 47 seconds to process a single frame and the data association method proposed in [12] can take up to 2 seconds to process a single frame. Combining these two methods, it may take hours for processing a 2-5 minute (1200 to 12000 frames) video on a general purpose computer (2.8

GHz CPU, 8 GB RAM).

In the era of Internet of Things, with CCTV cameras installed at numerous locations, big video datasets are getting routinely recorded for future analysis. On an average, the size of a 24 hours High-Definition (HD) video from CCTV cameras is approximately 70 GB (nearly 2500000 frames). Video analytics have been in use for real-time monitoring and for forensic analysis of scenes. In the case of large scale events or crime scenes, the video footage often provides evidence and clues for investigators. Forensic post analysis of massive amount of video data (often referred to as Big Data) using the existing MOT methods may take several days; hence, development of fast MOT methods is critically needed. This thesis proposes a novel greedy data association technique (GDA) which significantly improves the computational speed of MOT (at the data association stage) [15] and MapReduce-based techniques (MOT-MR) to parallelize the MOT methods to achieve computational speedup. The following section provides a high-level overview of the proposed solutions.

1.2 Proposed Solutions

The focus of this thesis is to improve the computational speed of MOT. To improve the computational speed of the data association stage, a Greedy Data Association technique (GDA) is proposed that does not use any image analysis or globally optimal techniques and provides qualitative performance comparable to the existing state-of-the-art methods. The central idea of GDA is an iterative matching of object detections in which the strictness of matching is decreased in each iteration. There are two key steps in GDA: Tracklet Creation step that links the object detections obtained from the object detection stage to form short reliable object trajectories/tracklets and Tracklet Association step that further

associates the short tracklets. Here, object trajectory of an object refers to the path travelled by the object in the video sequence. The use of the greedy algorithm to find the locally optimal solution allows proposed technique to be faster than the current state-of-the-art algorithms.

The proposed GDA technique improves the computational speed only for the data association technique. To improve the execution time of complete MOT methods that include both the object detection and data association stages, a parallel solution is proposed. A well-known distributed programming model, MapRedue [16], parallelizes the processing of large datasets by splitting the datasets into multiple chunks. The chunks of the given dataset are processed in parallel on multiple machines during the Map phase of MapReduce, and the intermediate results from the Map phase are combined in the Reduce phase. Using the MapReduce model, this thesis proposes MapReduce-based parallel techniques for MOT (MOT-MR). The MOT-MR techniques focus on improving the computational speedup by parallelizing the processing of a video file. A video file is split into multiple chunks and these chunks are processed independently during the Map phase of the MapReduce program. The intermediate results obtained from these chunks are combined to produce the continuous final object trajectories in the Reduce phase. Note that the continuous object trajectories refer to the object trajectories which are continuous for the complete video file. Processing multiple video files in parallel (each video processed on a single node) is a simpler task, however, processing a single video file in parallel is a non-trivial one (especially when time-dependencies are considered) because at the end, the results from all video chunks need to be combined for the subsequent sequential frames for the complete video file. This thesis focuses on processing a single video file in parallel. For handling the

time-dependencies in parallel processing of a video file, three MapReduce-based techniques have been introduced: the first technique distributes only time-independent tasks (at the object detection stage) on multiple computing nodes and performs time-dependent tasks (at the data association stage) sequentially; the second technique parallelizes both the time-dependent and the time-independent tasks; and the third technique extends the second technique by additionally using frame overlapping among the video chunks processed by different map tasks. The existing parallel solutions [17, 18, 19] focus on the architecture of distributed platforms for a general-purpose video analytic applications and parallelize the simultaneous processing of multiple small video files. These solutions do not provide low-level insights specific for parallel implementation of MOT for a single large video file and do not handle time-dependencies in MOT that occur when parallelizing the processing of a single video file. This thesis is directed at addressing the gap in the state-of-the-art methods based on parallel processing.

This thesis includes the various algorithms devised and describes a proof-of-concept prototype built for the MapReduce-based parallel MOT techniques. The prototype is deployed on an Amazon Web Services (AWS) Elastic Compute Cloud (EC2) and results of experiments investigating the performance of the proposed techniques are described.

MOT is a broad research field, and there are several distributed platforms available for parallelizing the processing of big datasets. The next section discusses the extent of this thesis and the specific technologies which this thesis focusses on.

1.3 Scope of the Thesis

The scope of this thesis straddles two areas: improving MOT and the use of parallel systems for MOT. The specific objective is to devise fast and scalable techniques for performing

MOT offline on stored video files. In the context of improving MOT techniques, this thesis specifically focuses on the data association technique and does not concern the object detection stage. For parallelizing the MOT methods, the MapReduce programming model is used and the proposed techniques are implemented on Apache Hadoop deployed on an AWS EC2 cloud. This thesis does not investigate other big data frameworks such as Apache Spark and Apache Storm in the context of MOT. Also, the performance analysis of the proposed techniques is carried out on general purpose CPU-based machines, and not on GPU-based machines. For performance analysis of the GDA technique, in this thesis, experiments have been performed by using video sequences having only homogeneous objects (such as pedestrians). The GDA technique can be easily extended to track heterogeneous objects (such as pedestrians and vehicles) because it does not utilize any object specific properties.

1.4 Limitations of the Thesis

Following are the limitations of this thesis:

- The accuracy of the proposed GDA technique decreases if large number of objects in a given video possess non-linear motion. For example, if a video is captured by a fast-moving camera then the GDA technique may not provide accuracy as good as when the video is captured using a stationary camera.
- Manual initialization of several parameters is required in the proposed GDA technique. The values of these parameters depend on various factors such as height of camera, angle of camera, speed of objects, and frame rate of a video.
- The proposed MOT-MR techniques assume that all the nodes of the cluster are homogeneous. The performance of MOT-MR techniques may vary if the

nodes of the cluster are heterogeneous. As an example, heterogeneous nodes may have CPUs with different clock frequencies.

1.5 Contributions of the Thesis

The main contributions of this thesis are:

- **A greedy data association technique**: A novel technique has been proposed for the data association stage of MOT. This technique finds locally optimum solutions which have similar qualitative performance as compared to the existing state-of-the-art methods and significantly improves the computational speed for data association.
- **MapReduce-based techniques for MOT**: Three MapReduce-based techniques have been introduced for MOT. The proposed techniques split the single video file and process the split video chunks on a multi-node MapReduce/Hadoop cluster and achieve significant speedup.
- **Proof-of-Concept Prototype**: A prototype executing on a Hadoop platform deployed on the Amazon EC2 cloud is built for demonstrating the efficacy of the proposed techniques.
- **Insights on System Performance**: Insights gained into system behavior and performance from the experimental evaluation of the prototype processing a number of benchmark video files are presented.

The following publications have resulted from this thesis work:

- G. Singh, S. Rajan, S. Majumdar "A Greedy Data Association technique for Multiple Object Tracking", in the Proceedings of IEEE Conference on Multimedia Big Data, 2017.

- G. Singh, S. Majumdar, S. Rajan "MapReduce-based Techniques for Multiple Object Tracking in Video Analytics", in the Proceedings of IEEE Conference on Cloud and Big Data, 2017 (accepted for publication).

1.6 Outline of the Thesis

The next chapters of this thesis are briefly outlined: Chapter 2 discusses the background information related to MOT and the MapReduce programming model and provides a discussion of the existing work related to the proposed techniques. The proposed GDA technique and the performance analysis of GDA have been discussed in Chapter 3. The MapReduce-based techniques and the performance analysis of these techniques are discussed in Chapter 4. Finally, Chapter 5 concludes this thesis and discusses the future work.

Chapter: 2 Background and Related Work

This chapter discusses the background concepts of MOT, the MapReduce model, Apache Hadoop, and the EC2 cloud provided by AWS. The existing research related to the proposed techniques in this thesis is also discussed. Discussion of the background concepts is presented in the following sections. Section 2.1 provides background information on MOT while Section 2.2 and Section 2.3 focuses on MapReduce and related technologies. Finally, Section 2.4 discusses the related work.

2.1 Multiple Object Tracking

In the last century, MOT was used for tracking aircraft by estimating the target states using a Kalman filter [20]. The major tracking applications were air traffic control, ocean surveillance, battlefield surveillance and ballistic missile defense. The targets were detected with the help of radar signals. During the last two decades, MOT has been applied for tracking the visual objects in the sequence of images (video sequences). Fig. 2.1 shows the examples of MOT in two video sequences. The main idea is to assign a unique identity

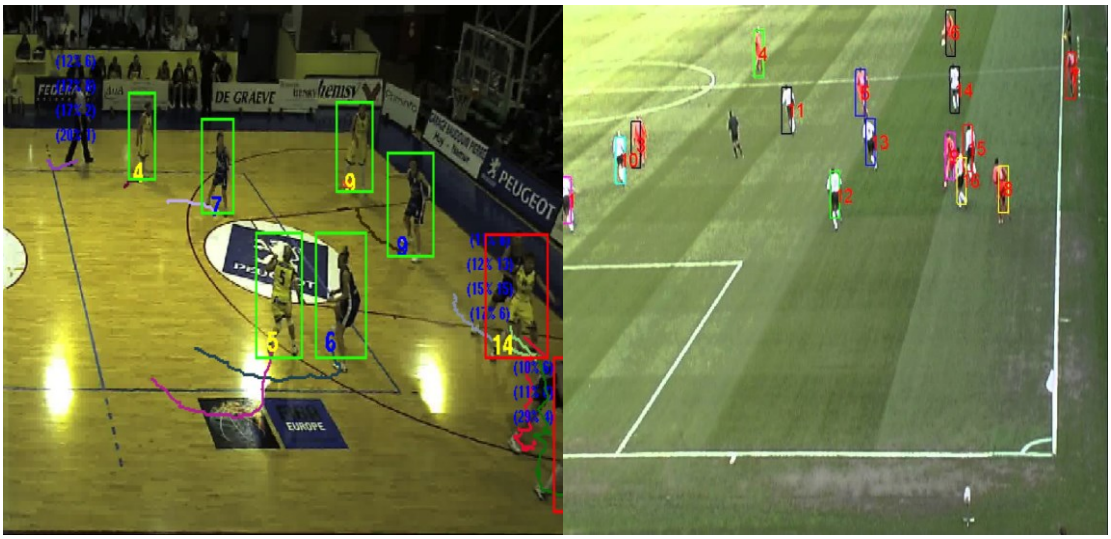

Figure 2.1: Examples of multiple object tracking in video sequences. Source [65, 66]

to an object in the video sequence and then track that object with the same identity in the different frames in the video sequence. An object of interest in the image is usually represented by a rectangular bounding box. If the object moves out of the frame and re-enters the frame, then it is considered as a new object.

2.1.1 Affinity Measures

Affinity measures are used for calculating the cost/distance of matching between two objects or object trajectories. The most common affinity measures used for MOT are based on the appearance model and the motion model. The appearance model is used for comparing the appearances of objects across the frames. Appearance model describes a model for extracting the characteristics and features of an object and comparing these characteristics with other objects and with the background scene. There are two goals of the appearance model. First, it differentiates the object appearance from other objects, so that two different objects are not associated with the same trajectory. Second, it discriminates the object from background scene. Examples of the appearance based-affinity measures are linear logistic regression classifier [21], discriminative part-based appearance model [22], aggregated local flow descriptor [23], and Kullback-Leibler distance [24].

The motion model is used to predict the future position of an object by learning the motion pattern of an object. The learned motion patterns are then utilized for associating the object trajectories. Motion model reduces the sample space by limiting the motion of an object within the real-world possibilities. For instance, there is an upper limit to the velocity with which an object can move, which restricts the area of search in the image for locating the

objects' locations. Affinity measures based on the motion model such as linear object motion [25] and non-linear object motion [13] are often used in MOT.

2.1.2 MOT Based on Manual Initialization

Given a video file as an input, the process of tracking a single object (or region of interest) always requires manual initialization of the object at the start of the video sequence. Manual initialization refers to marking/initializing the location of the object to be tracked in the first frame. In the consecutive frames, the initialized object is tracked by utilizing the object appearance, and by learning motion of the object. Manual initialization-based tracking is largely based on the complex image comparison algorithms since the key part is to locate the initialized portion of the frame in subsequent frames. Some existing single object tracking methods based on manual initialization are kernel-based tracking [26], fragments-based tracking [27], and mean shift tracking [28]. These methods utilize histogram matching for locating the initialized object across the frames.

The manual initialization-based MOT methods also exist in the literature. Examples of the manual initialization-based MOT methods are incremental Log-Euclidean Riemannian subspace algorithm for appearance comparison [29], object structural relations [30], and object joint motion estimation [31]. The manual initialization-based tracking has one major drawback. Only the objects which are initialized at the start of the video are tracked, whereas, other objects, the ones that enter the scene after the first frame, are not tracked.

2.1.3 Two-stage Multiple Object Tracking

Availability of large benchmark datasets on the web and accelerated computing hardware has propelled the research in artificial intelligence and signal processing. Computers can perform the tasks which only humans could do, and one such example is an object

detection. Object detection is a process of finding an instance of a certain object in each image. Various applications of the object detection include face detection, pedestrian detection, animal detection and vehicle detection. Several state-of-the-art object detection algorithms [7, 8, 32, 9] were proposed during the last decade which enhanced the object detection accuracy. With the enhancement of the object detection algorithms, most of the existing MOT methods consist of two stages: the object detection stage and the data association stage. The object detection stage is a time-independent process of obtaining spatial locations of objects (object detections) available in a video sequence whereas the data association stage is a time-dependent process of associating the obtained object detections into the probable object trajectories [6]. The two stages of MOT are sometimes collectively termed as Tracking-by-Detection (TBD). Fig. 2.2 shows an overview of the two-stage MOT method. Pre-trained detectors are generally used to detect an object, and there is no frame to frame dependencies in this stage. The object detector detects the object in each frame independent of any other frame. The data association stage associates the

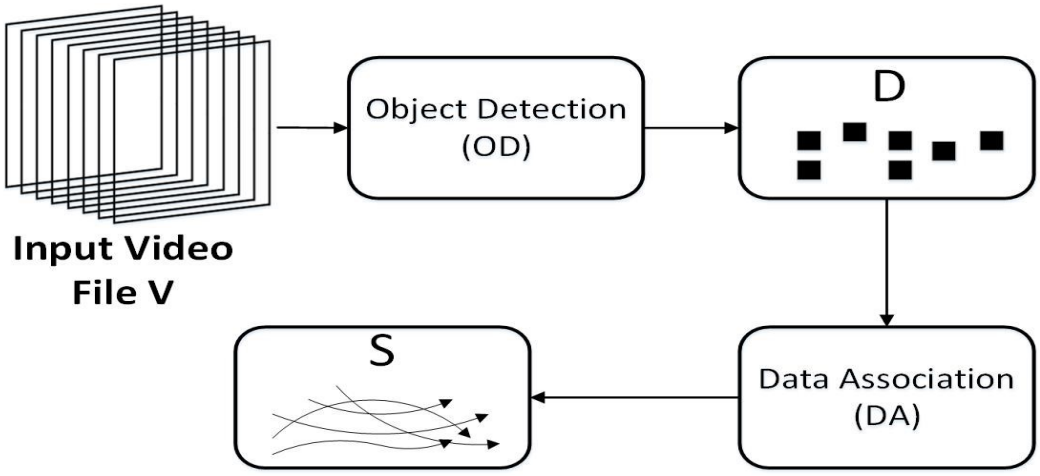

Figure 2.2: The two-stage multiple object tracking method. D is a set of object detections, and S is a set of final object trajectories.

obtained object detections to generate continuous object trajectories of the objects available in the video sequence.

The two-stage MOT overcomes the drawback of manual initialization of objects in the starting frame. In two-stage MOT, all the objects are detected by object detector. As a result, unlike manual initialization method, every object of interest is tracked, even those entering the frame after the start of the video sequence. In this manner, a variable number of objects can be tracked. Since this method has two stages, it is important to note that qualitative and computational performance of MOT techniques are dependent on both the stages.

2.1.4 Online vs Offline Tracking

Tracking of an object can be either online (real-time) or offline (batch). These two ways of tracking differ in the use of future frames. Online object tracking examines current and past frames only, whereas offline tracking considers all the frames, and performs tracking as a batch processing. Online tracking is used for real-time processing, especially for live stream videos. Examples of online tracking include tracking in an autonomous vehicle, tracking by a robot and or tracking of sports players during a live match. For online tracking, a complete video is not available before the tracking could be started. On the other hand, offline tracking usually deals with post analysis of the video; hence, the entire video is available before starting the tracking, and for processing current frame, both past and future frame observations are used. The proposed techniques in this thesis focus on offline tracking.

2.2 MapReduce

In 2004, Google introduced a MapReduce programming model [16]. This programming model simplifies the development of distributed and parallel programs that enable the processing of big data sets in a timely manner. The parallel programs can be written by using a number of available programming languages. The MapReduce programming model provides a level of abstraction to hide all the complex implementation details which are required to be dealt while writing the distributed programs. Fig. 2.3 shows an overview of MapReduce model using an example. All the processes in a MapReduce job are implemented in two key phases: Map phase and Reduce phase, and the intermediate results between these two phases is transferred in the Shuffle and Sort phase. The processes which can be completely parallelized are implemented in the Map phase. The Reduce phase usually combines the output from the Map phase. The data to be processed is split into

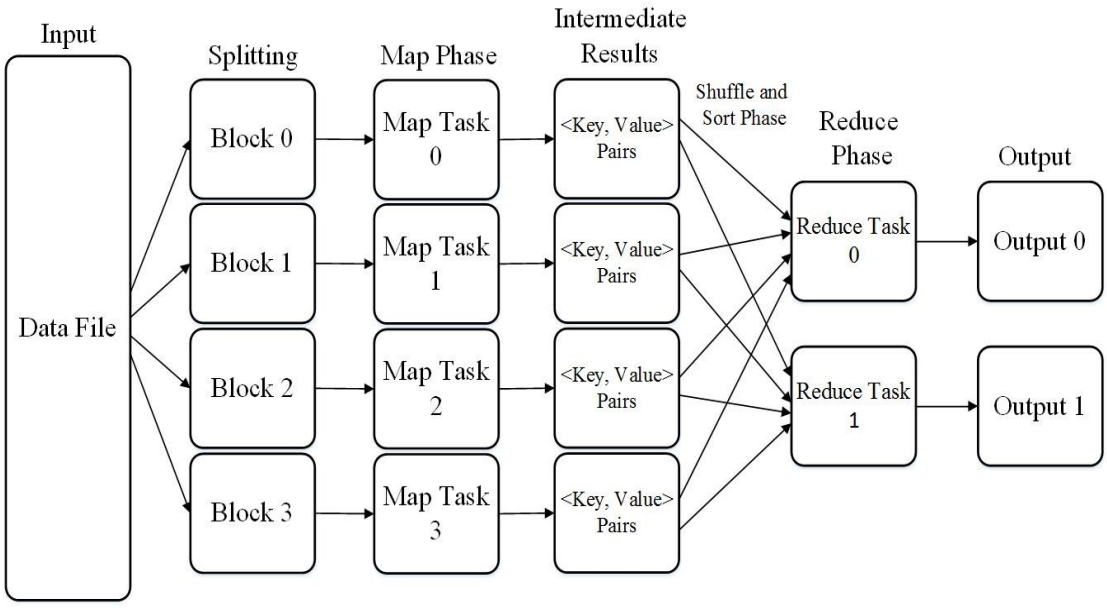

Figure 2.3: Map and Reduce phase of an example MapReduce application

multiple blocks (four in the example in Fig. 2.3). Each block is processed independently by a single map task during the Map phase. Note that the number of map tasks executed during the Map phase is always equal to the number of blocks of data (see Fig. 2.3). The output of Map phase is intermediate results in the form of <key, value> pairs. These intermediate results are sent to the reduce tasks during the Shuffle and Sort phase. The Shuffle and Sort phase transfers the intermediate <key, value> pairs based on the key of each <key, value> pair. The pairs having the same key are sent to the same reduce task. The reduce task processes these intermediate results and generate the final results which are also in the form of <key, value> pairs. By default, the number of reduce tasks is one in Apache Hadoop, and it can be varied depending on the problem to be solved.

2.2.1 Apache Hadoop

Apache Hadoop is a popular open-source implementation of the MapReduce programming model. It uses a master-slave architecture. A user submits a job to the master node, and then the master node distributes the tasks in the job among the various slave nodes. The underlying file system of Apache Hadoop is called Hadoop Distributed File System (HDFS) which is based on the Google File System proposed in [33]. The data uploaded on HDFS is divided into multiple blocks, each of size 128 MB by default. The multiple blocks of data are distributed among the various slave nodes. The master node keeps a record of all the metadata which includes directory namespaces, mapping of the chunks to the slave nodes and data replication. The Apache Hadoop framework provides fault tolerance by replicating data on multiple slave nodes and enhances runtime efficiency by maximizing data locality. When the Hadoop job is executed, the Hadoop framework tries to schedule

the processing of each chunk on its local slave node to provide data locality. The detailed explanation of Apache Hadoop is given in [34].

2.2.2 HDFS

HDFS is a java based distributed filesystem which runs on the top of a local file system. It is based on Google's Distributed File System. It provides storage space for a vast volume of data. The actual application data and metadata are stored separately on HDFS. The metadata is stored on Namenode (master) whereas actual application data is stored on multiple Datanodes (slaves). All the nodes of HDFS cluster are connected by high bandwidth network and communication is based on the TCP protocols. The three important components of HDFS are Namenode (master), Datanode (slave) and HDFS Client.

- Namenode: It is a master component of HDFS, which keeps a record of all the file storage processes. Namenode saves all the metadata of application data. It keeps attributes like permissions, access times, available storage in a hierarchy of file and directories. Whenever user application requests a file read/write operation to Namenode, it provides the user application the actual data storage location of the file in the Datanodes.
- Datanode: A Datanode is used to save the actual user data. On submission of a new data file by the user, the file is divided into chunks of 128 Mb (by default) and these chunks are stored on multiple nodes. Each chunk of data is replicated three times by default. The data replication provides fault tolerance and reliability.
- HDFS Client: It is a library that provides an interface between user applications and HDFS. The user application can perform simple operations

such as read, write, or delete files and create or delete directories through this HDFS Client.

2.2.3 YARN (Yet Another Resource Negotiator)

YARN is a new resource manager for Hadoop which was introduced in Hadoop version 2. The basic idea of YARN is to divide the task of resource management and job scheduling. It uses three daemons named ResourceManager, NodeManager and ApplicationMaster.

- ResourceManager: It executes on the Master node of Hadoop Cluster and performs the resource management. The main task of ResourceManager is to split the resources among different MapReduce applications. There are two components of ResourceManager: Scheduler and ApplicationManager. Scheduler allocates the resources to the applications running on the Hadoop cluster, whereas ApplicationManager monitors the applications. Whether to accept an application or restart a failed application is a task of ApplicationManager. The default scheduler used in ResourceManager is Hadoop's Capacity Scheduler, whereas other options such First In First Out (FIFO) and Fair scheduler are also available.
- NodeManager: YARN executes NodeManager on each of the slave nodes to monitor the resource usage, and provides feedback to the ResourceManager.
- ApplicationMaster: It is executed on a slave node. For each application, YARN runs one ApplicationMaster. The task of ApplicationMaster is to negotiate for resources with ResourceManager. In conjunction with NodeManager, it also monitors the tasks of its application running on the slave nodes.

2.3 Cloud Computing

MapReduce applications are often executed on clouds. Cloud computing provides shared computing resources to the users over the internet. It is an On-Demand service, where resources can be acquired as and when required. One of the major advantages of cloud computing is a pay-as-you-go service that allows users to pay only for the resources used. Other useful characteristics of cloud services are agility, cost reduction, scalability, and elasticity. Since the last decade, cloud services emerged as one of the greatest boons to the computer technology, in particular for the small-scale entrepreneurs and start-up companies. There are three types of service models provided by cloud computing service providers.

- Software as a Service: It provides users access to various software applications which are running on cloud infrastructure.
- Platform as a Service: It allows the users to use the cloud infrastructure for deploying the user developed applications.
- Infrastructure as a Service: It is one of the most basic cloud services, which provide a user the access to use the basic infrastructure such as a virtual machine in the cloud.

The various multi-national companies such as Amazon, Google, Microsoft and Oracle provide cloud computing services. The research work proposed in thesis uses the Elastic Compute Cloud (EC2) provided by Amazon Web Services (AWS). The reason for using EC2 from AWS is the ease and simplicity with which the resources can be accessed on the cloud. AWS provides Command Line Interface (CLI), a tool to manage the AWS services.

2.4 Related Work

This thesis concerns data association techniques and parallel processing for improving the computational speed of MOT. A representative set of existing research in each of these areas is presented. The related work on the MOT techniques is discussed first (see Section 2.4.1 and Section 2.4.2) and the discussion of parallel processing and MOT is presented next (see Section 2.4.3).

2.4.1 Object Detection for MOT

With the development of accurate detection algorithms, high-speed CPU, and complex image processing techniques, tracking-by-detection paradigm (two-stage) has been adopted by most of the researchers where MOT is formulated as a data association problem after the object detections are obtained. Pre-trained detectors provide detection hypotheses, the independent object detections for each frame, which form the input for the data association problem. At the object detection stage, background subtraction was successfully used in [35, 36] for detecting the moving targets independently in each frame. Background subtraction is a process of extracting the image foreground for further processing. The background subtraction method works only for stationary cameras where the background of the frames in a video sequence remains alike. In the case of moving cameras, background subtraction does not work efficiently because the background and foreground (objects) of the frames are moving continuously.

The object detection algorithms proposed in [7, 8, 9, 14, 32, 37, 38] make use of machine learning-based techniques. These object detectors produce more accurate detection hypothesis, especially by classifying the detections with a class label such as a vehicle, pedestrian, and animal. The existing state-of-the-art object detection method,

Discriminatively Trained Deformable Parts Model (DPM) in [38] uses multi-scale deformable parts model where the objects are modeled by visual grammars. The method in [38] builds an object by combining multiple parts of the object where each part of the object is a grammar model which can be defined directly or as a collection of the other parts of the object.

Several papers that investigate the feasibility of using deep learning techniques have been recently published [37, 39, 40, 41, 42]. In [14], a region-based convolutional network (R-CNN) was proposed. R-CNN creates bounding boxes for detected objects in the following manner. It first generates potential bounding boxes in a given image. Then it runs a classifier on the proposed boxes. A post-processing is then done to refine the bounding boxes to eliminate any possible duplications in detections. These processes are slow and hence to improve the speed and detection accuracy, a fast region-based convolutional network was proposed in [37]. Also, further improvement in detection performance was demonstrated by using local and contextual information in [39] that used fast R-CNN. The network in [39] used bi-directional networks to pass messages between features and used gating functions to control the messages. To improve speed, a single convolutional network that simultaneously predicts bounding and classifies bounding boxes was proposed in [41]. This was termed as "you look only once" (YOLO) approach. YOLO approach was a fast real-time approach for detection of objects.

In conclusion, machine learning-based object detection algorithms have significantly improved the detection accuracy in comparison to background subtraction, and these methods work for both the stationary and moving cameras. Thus, these machine learning-based object detection methods have become favorable choices for MOT.

2.4.2 Data Association for MOT

Recently, several offline data association techniques have been proposed for MOT and most of the proposed techniques use global optimization. For instance, [43] has formulated the multi-target tracking as a multipath searching problem and has produced a globally optimized solution by using linear programming. However, the method proposed in [43] requires the total count of targets to be known *a-priori***.** Data association method presented in [11] formulated MOT as a Minimum Cut Subgraph Multicut Problem (JMC) and solves for the globally optimal solution. A 3D scene layout was used in probalistic-based generative model for traffic scene understanding (referred to as TBD) in [44]. These methods solve for globally optimal solutions, and thus, have a high computational cost.

Formulating network model from given detection hypothesis for data association is another approach in offline data association methods [6, 45, 46, 47]. The k-shortest path optimization is used to solve the network model in [6]. A greedy algorithm for non-max suppression (DPNMS) is used in [45] to remove false detections and the k-shortest path algorithm is used for associating targets. The method in [45] is successful in achieving fast processing speed, but non-maximally suppressing the detection responses led to a large number of false negatives. Unlike [45], the proposed GDA technique is used for the actual association of tracklets and not for suppressing the detection responses. Moreover, authors of [45] associate one object trajectory at a time by using the shortest path algorithm, whereas the GDA technique proposed in this thesis grows all trajectories simultaneously in multiple iterations.

2.4.2.1 Iterative Data Association for MOT

A group of offline tracking methods [10, 12, 13, 48] uses iterative algorithms to solve the data association problem. One of the advantages of using iterative algorithms is that with each iteration as some of the object trajectories are associated, the solution space reduces. Thus, many of the difficult-pairs of object trajectories get associated more accurately in the later iterations. Following this approach, the method presented in [12], tracking by continuous energy minimization (CEM), formulated the data association problem as the minimization of continuous energy where energy functions considered several aspects such as target dynamics, physical constraints, mutual exclusion, and histogram based appearance model. Final object trajectories are produced after several iterations. Although formulating an optimization problem by using multiple energy functions leads to qualitative performance improvement, it also increases the runtime by a large factor. Tracking by online learning of non-linear motion of objects is introduced in [13] where the Hungarian algorithm is used for providing a globally optimal solution. Note that the Hungarian algorithm solves the assignment problem by using a combinatorial optimization technique and takes polynomial time [49]. The method proposed in [13] improves the tracking results by tackling abrupt object motions. However, learning of non-linear object motion and use of the Hungarian algorithm make the method in [13] a computationally slow process. The iterative method in [48] tracked multiple objects by using a hierarchical association of the object detections. A dual threshold-based pre-processing stage is used to create reliable tracklets by associating safe detections and then the Hungarian algorithm is used to further associate the reliable tracklets to provide the final trajectories. The GDA technique proposed in this thesis also uses a similar dual threshold-based approach in the Tracklet Creation step,

however, instead of the Hungarian algorithm followed by a pre-processing step, GDA uses threshold-based approach for further data association as well, which enables GDA to be computationally fast. Further, instead of using a fixed value of the threshold, GDA iteratively increases the threshold value to reduce the strictness of matching after each iteration. Initially, a lower threshold value is used to associate the safe pairs (pairs which have low association cost) of tracklets while increasing threshold enables the association of difficult pairs (pairs which have high association cost) of tracklets in subsequent iterations when solution space reduces in every iteration.

Multiple hypothesis tracking (MHT), earlier introduced in [20], is reused in [50] by additionally learning the online appearance models for addressing the varying object appearances. Note that the actual tracking method is offline, only appearances are learned online by using features from deep convolutional neural networks. Although MHT [50] achieved a high performance, it was at the expense of a high processing cost. This thesis proposes a novel GDA technique that does not use appearance model and performs iterative object matching only based on a linear motion model to achieve computationally fast results. The safe pairs of detection responses are associated before the uncertain difficult pairs. The core idea of GDA is an iterative matching by decreasing the strictness of matching in each iteration. The proposed GDA technique improves the computational speed at the data association stage of MOT while producing qualitative performance comparable to the current state-of-the-art algorithms.

The use of proposed greedy technique and other greedy algorithms such as [45] used in the data association stage improve the computational speed only for the data association stage. To improve the computational speed of the object detection stage, the use of deep learning

algorithms deployed on Graphics Processing Units (GPUs)-based systems have been proposed in [37, 51]. These methods can reduce the runtime of the object detection stage to some extent (0.2 to 10 seconds per frame). However, these GPU-based systems are more expensive when compared with general purpose computers.

2.4.3 MapReduce/Hadoop-based Parallel Techniques for Video Processing

A continuous increase in the production of large video data sets and computationally slow algorithms that are used currently for the analysis make parallel and distributed platforms a fast-growing research area for video processing applications. During the last five years, a number of methods [17, 18, 19, 52, 53, 54, 55] have been presented on parallel and distributed platform-based applications ranging from video storage, video encoding, video transmission and video surveillance etc. For instance, the authors of [52] have proposed a Hadoop file system (HDFS)-based architecture for efficient video storage. Similarly, Hadoop-based high-speed video encoding and decoding methods were proposed in [53, 54].

A limited number of solutions running on Hadoop, a well-known platform for running MapReduce Applications, have been proposed for video analytics applications. Parallel processing of image databases using MapReduce is proposed in [18, 25], however, the proposed technique parallelizes only the image processing application and does not handle MOT for video files. A Hadoop-based video processing framework is proposed in [17]. It uses a cloud environment and demonstrates speedup by running a demo program of face detection (no time-dependencies) using an Opencv-based image processing software. The use of cloud services for video analytics-based traffic monitoring is proposed in [19]. It provides a model for storing video stream from traffic monitoring cameras to HDFS and uses the Opencv library to perform video analytics-related jobs. However, this method

stores multiple small size videos and processes each of the stored videos independently. Another Hadoop-based video analytics solution is given in [57] and speedup is demonstrated using multiple small video files. Although the method in [57] also splits a single video file for parallel processing, it does not effectively utilize the Map and Reduce phases of MapReduce for the video processing operations. Instead of utilizing the Map phase for parallelizing the operations, this method uses a single map task for splitting a video file and only uses Reduce phase for video processing operations.

The Hadoop-based methods discussed, so far, focus on using the Hadoop architecture for storing and processing general video analytics applications, however, these methods do not describe parallel processing techniques required for the handling of the end-to-end MOT process, and do not handle time-dependencies associated with MOT. In this thesis, to address the shortcomings of the existing literature, three MapReduce-based techniques for implementing MOT have been introduced. The proposed MOT-MR techniques that are discussed in Chapter 4 handle time-dependencies associated with distributing a single video file on a multi-node Hadoop cluster.

Chapter: 3 Greedy Data Association Technique

This chapter discusses a greedy data association technique (GDA) for MOT introduced in this thesis. Section 3.1 presents the major symbols and notations used and Section 3.2 gives an overview of GDA. The main idea is presented in Section 3.3. Section 3.4 and Section 3.5 discuss the two key steps, namely Tracklet Association and Tracklet Creation, of the proposed technique. Linear Motion Cost Model used for data association is discussed in Section 3.6, and Section 3.7 gives details of input data filtration. Finally, the performance analysis of the GDA technique using the benchmark video datasets is given in Section 3.8.

3.1 Symbols and Notations

GDA is a data association technique which associates the object detections obtained from the object detector. The major symbols and terminologies used in this method are first introduced. The input detection hypothesis, denoted by $D = \{d_i^{f_i}\}$, obtained in the object detection stage is a set of all the detections in a video sequence where the i^{th} object $d_i^{f_i} = \{x_i^{f_i}, y_i^{f_i}, w_i^{f_i}, h_i^{f_i}, conf_i^{f_i}\}$ is defined by rectangular bounding box with $(x_i^{f_i}, y_i^{f_i})$ as the center and $(w_i^{f_i}, h_i^{f_i})$ as the width and the height, $conf_i^{f_i}$ as the confidence score and f_i as the frame number. An object trajectory/tracklet $T_k = \left\{ d_k^{f_k^s}, d_k^{f_k^s+1}, \dots, d_k^{f_k^e} \right\}$ is an ordered set of detections in consecutive frames, representing the continuous trajectory of a single object, where each object detection $d_k^{f_k} \in D$. Also, f_k^s is the starting frame and f_k^e is the ending frame of the tracklet T_k. A set $S = \{T_1, T_2, \dots, T_L\}$ is a set of tracklets where L is a total number of tracklets. Associating two tracklets T_i, T_j forms a single tracklet T_k such that $T_k = \{T_i, U_{ij}, T_j\}$ represents trajectory of one object. For association, the end frame f_i^e of tracklet T_i must be less than the starting frame of f_j^s of tracklet T_j. Two associating tracklets

can possibly have a temporal gap ($f_j^s - f_i^e > 0$), where object detections are missing due to the object being occluded or detector being unable to detect the object. For example, if the tracklet T_i ends at the frame number 100 and the tracklet T_j starts at the frame number 105, then the detections of the object in the frames 101-104 are missing. A set of detections, U_{ij}, is created by GDA between the end of T_i and start of T_j, which are originally missing from the available input detection hypothesis. These missing detections are created by calculating average positions based on linear velocity between last detection of T_i and the first detection of T_j. Here, the centroid of the bounding box, which is given by $(x_i^{f_i}, y_i^{f_i})$, is used as a position of $d_i^{f_i}$.

3.2 An Overview of GDA

Fig. 3.1 shows an overview of the GDA technique. The object detection method is applied to the input video file, and a set of object detections, D, is obtained. These object detections are then passed to the GDA technique. There are two key steps in GDA: Tracklet Creation and Tracklet Association. The pre-data filtration and post-data filtration steps are used for removing the incorrect object detections (false positives) from the given input detections. These data filtration steps are discussed in detail in Section 3.7. The Tracklet Creation step associates the object detections based on the distance between objects and creates short tracklets. These short tracklets are further associated in the Tracklet Association step. The Tracklet Creation and Tracklet Association steps are discussed in detail in Section 3.4 and Section 3.5 respectively.

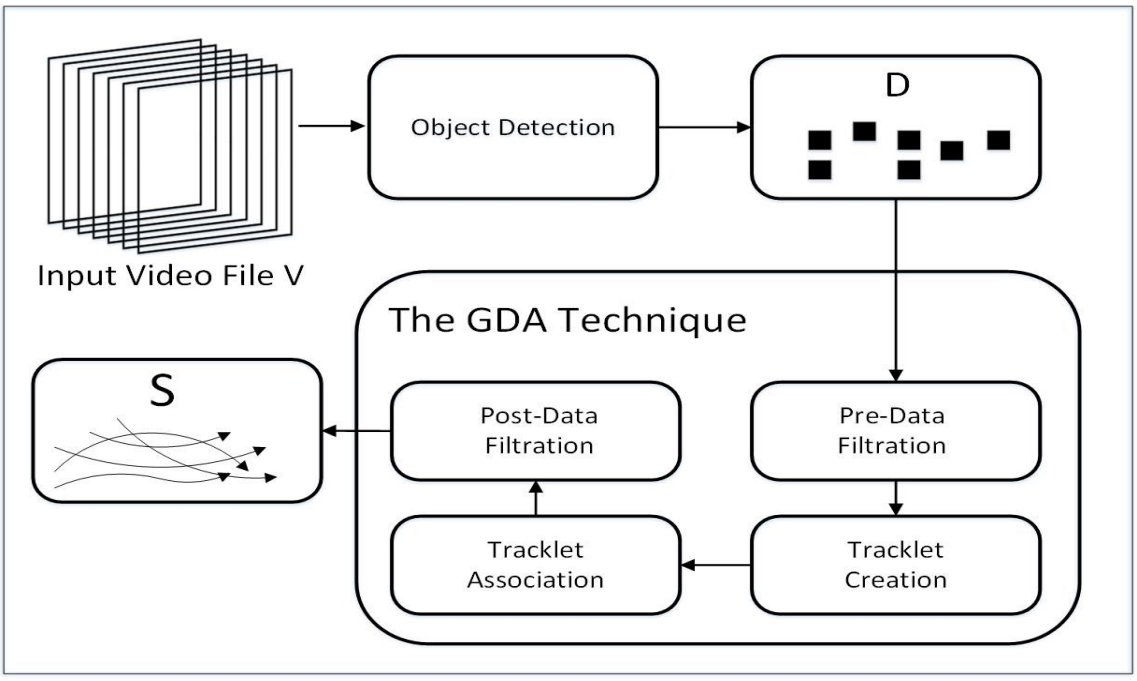

Figure 3.1:An overview of the Greedy Data Association technique.

3.3 Main Idea

The main idea of the proposed data association technique, GDA, is illustrated in Fig. 3.2. Fig. 3.2(a) shows given input of object detections over a sequence of frames, 0 to 11, where each object detection is shown by a blank circle. The Tracklet Creation step is used to associate the given object detections to create initial short tracklets. A set of tracklets, S^0, is the output of the Tracklet Creation step. Tracklet Creation is shown by transition from Fig. 3.2(a) to Fig. 3.2(b). In Fig. 3.2(b) each set of detections with same the color represents a unique tracklet. In the Tracklet Association step, the short tracklets are further associated by using multiple iterations, as shown in Fig. 3.2(c) and 3.2(d).

The use of an iterative algorithm for Tracklet Association is motivated by two important observations. First, with each iteration, as some of the tracklets get associated, the total

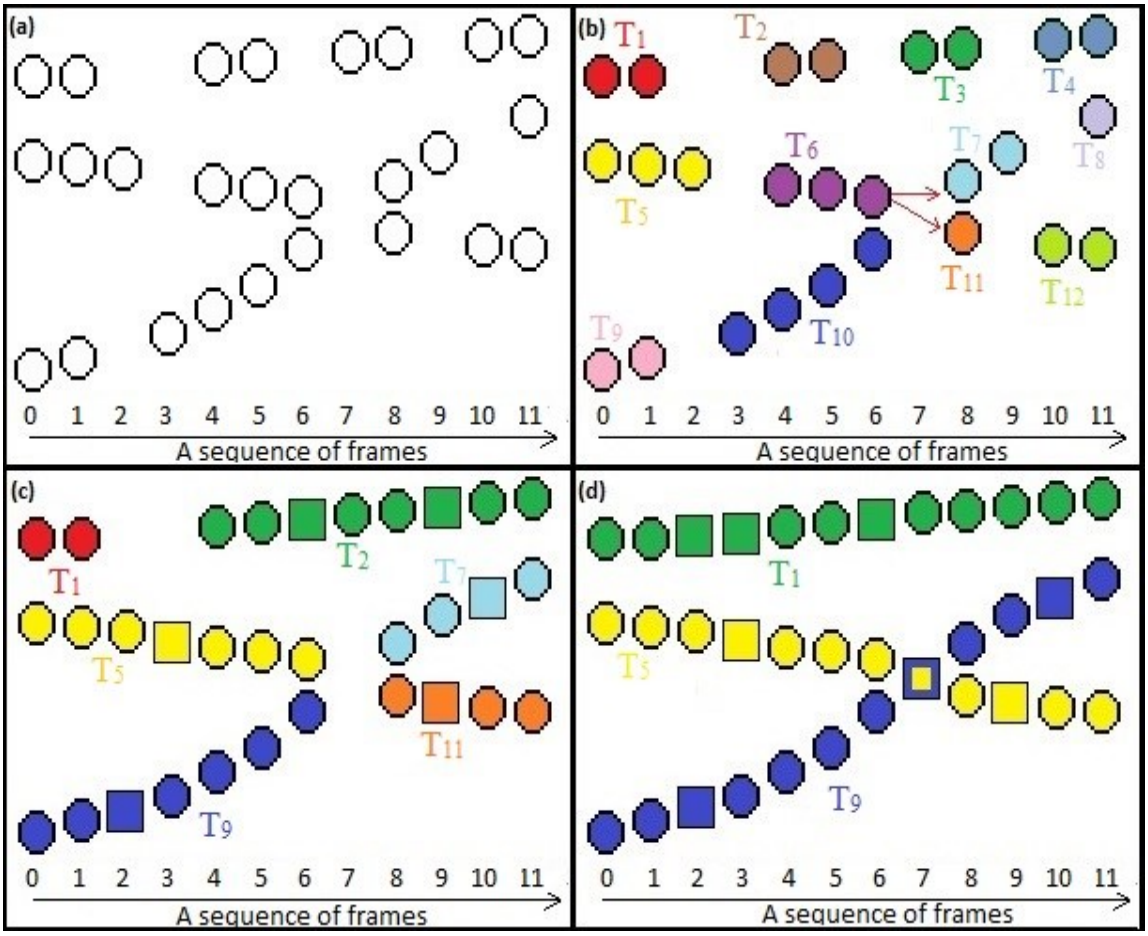

Figure 3.2: Main idea of the Greedy Data Association technique: (a) Input Object Detections; (b) Tracklet creation; (c),(d) Tracklet association. Figure is shown best in color.

number of tracklets passed to the next iteration is reduced (solution space gets reduced). Second, as tracklets are associated in each iteration, the length of tracklets grow longer, yielding more accurate information about the motion of an object. Both observations are demonstrated by the transition shown in Fig. 3.2(b) to Fig. 3.2(c). As given in Fig 3.2(b), there is a total of 12 tracklets, and with association among these tracklets, the number of tracklets is reduced to 6 in Fig 3.2(c). Based on the two observations, Tracklet Association is started by only associating the pair of tracklets with significantly lower association cost, i.e. by keeping the highest strictness level. In succeeding iterations, as the tracklets grow longer and the solution space is reduced, the difficult pairs of tracklets are associated more

accurately, as shown by the transition between Fig 3.2(c) and 3.2(d). A difficult pair of tracklets may have one of the following characteristics: multiple possible predecessors or successors, higher association cost, a small length that does not convey enough motion information, or large frame gap. For instance, both tracklets T_6 and T_{10} in Fig. 3.2(b) could associate with tracklets T_7 or T_{11}. But in Fig. 3.2(c), after some possible associations, there is a higher certainty of tracklet T_5 associating with tracklet T_{11}, and tracklet T_9 associating with tracklet T_7. Also note that in Fig. 3.2, square boxes represent the estimated detections created by GDA which were originally missing from input detection hypothesis. Fig. 3.2(d) shows the final trajectories as T_1, T_5, and T_9.

3.4 Tracklet Association

Here, the main objective is to associate a set of short tracklets, S^0, obtained from the Tracklet Creation step defined in Section 3.5. Here, S^0 represents a set of initial tracklets which are associated iteratively in the Tracklet Association step. For each iteration, a set of tracklets $S^i = \{T_1^i, T_2^i, \dots, T_{L_i}^i\}$ is input (starting with i=0) and after data association is performed, the output set of tracklets $S^{i+1} = \{T_1^{i+1}, T_2^{i+1}, \dots, T_{L_{i+1}}^{i+1}\}$ is produced where each $T_j^{i+1} = \{T_1^i, U_{12}^i, T_2^i, \dots, T_{(n-1)}^i, U_{(n-1)n}^i T_n^i\}$ is a tracklet formed by associating some of the tracklets from the previous iteration. Also, U_{12}^i is a set of detections created by GDA between the tracklets T_1^i and T_2^i, which were originally missing from the input detection hypothesis. Similarly $U_{(n-1)n}^i$ is a set of detections created between the tracklets $T_{(n-1)}^i$ and T_n^i. It is important to note that, for each iteration, more than two tracklets could associate to form a single tracklet. For instance, tracklet T_2 given in Fig. 3.2(c) is formed by associating three tracklets T_2, T_3, and T_4 in Fig. 3.2(b). The tracklets which do not associate with other tracklets, are also passed on to the next iteration.

Any two tracklets T_j^i, T_k^i are considered overlapping if the succeeding tracklet T_k^i starts before the preceding tracklet T_j^i ends. The overlapping of tracklets indicates that these two tracklets belong to two different objects and cannot be associated. Thus, two tracklets ($T_j^i \rightarrow T_k^i$) are considered for association only if the tracklets are non-overlapping, that is,

$$0 < f_k^s - f_j^e < G_{max} \tag{3.1}$$

Here, f_k^s is the starting frame of the tracklet T_k^i, and f_j^e is the last frame of the tracklet T_j^i. Also, T_j^i is the succeeding tracklet of the tracklet T_k^i, and T_j^i is the preceding tracklet of the tracklet T_k^i. In Eq. (3.1), G_{max} denotes the maximum temporal gap (number of frames) between two tracklets that can be associated. The pair of tracklets are associated greedily, based on the linear motion cost model which is defined in Section 3.4, and the association cost between pair of tracklets ($T_j^i \rightarrow T_k^i$) is denoted by:

$$A_{COST}(T_j^i, T_k^i) = \begin{cases} \varphi(T_j^i, T_k^i) & \text{if (3.1) is true} \\ \infty & \text{otherwise} \end{cases} \tag{3.2}$$

The pair ($T_j^i \rightarrow T_k^i$) is associated if its association cost is:

(i) less than threshold θ_1^A:

$$A_{COST}(T_j^i, T_k^i) < \theta_1^A \tag{3.3}$$

(ii) less than association cost of other combinations of T_j^i with its succeeding tracklets given by ($T_j^i \rightarrow T_x^i$), where θ_2^A is another threshold:

$$A_{COST}(T_j^i, T_x^i) - A_{COST}(T_j^i, T_k^i) > \theta_2^A, \ \forall \ T_x^i \in S^i - \{T_j^i, T_k^i\} \tag{3.4}$$

(iii) and less than association cost of other possible combinations of T_k^i with its preceding tracklets given by $(T_x^i \rightarrow T_k^i)$:

$$A_{COST}(T_x^i, T_k^i) - A_{COST}(T_j^i, T_k^i) > \theta_2^A, \quad \forall\, T_x^i \in S^i - \{T_j^i, T_k^i\} \tag{3.5}$$

Otherwise, the pair $(T_j^i \rightarrow T_k^i)$ is not associated and other possible combinations of tracklet pairs are checked. This association of tracklets is not globally optimal since it does not solve for minimizing the global association cost of all the tracklets over a sequence at once. But it certainly prevents the erroneous associations by starting with a lower value of threshold θ_1^A, which keeps the matching strictness very high. In each iteration, every tracklet pair that satisfies Eq. (3.3), (3.4), and (3.5) is associated and a locally optimal solution is found. It should be noted that any tracklet T_j^i could associate with at most one preceding tracklet T_P^i, and at most one succeeding tracklet T_S^i. Within each iteration, this condition reduces the search space. For example, if a pair of tracklets $(T_j^i \rightarrow T_k^i)$ is associated, other possible combinations such as $(T_j^i \rightarrow T_x^i)$ and $(T_x^i \rightarrow T_k^i)$ are ruled out where $T_x^i \in S^i - \{T_j^i, T_k^i\}$. Any number of tracklets with continuous associations can form a new tracklet. For instance, if the pairs $(T_x^i \rightarrow T_y^i)$ and $(T_y^i \rightarrow T_z^i)$ are associated, then the newly formed tracklet is given by $T_j^{i+1} = \{T_x^i, U_{xy}^i, T_y^i, U_{yz}^i, T_z^i\}$ where $T_j^{i+1} \in S^{i+1}$. Thus, the set S^{i+1} is obtained which becomes input for the next iteration.

Algorithm 3.1 shows the Tracklet Association step of the GDA technique. The input is a set of tracklets which is obtained from the Tracklet Creation step. The iterative data association starts at Line 2, where the threshold θ_1^A (superscript A refers to the Tracklet Association step) is increased after every iteration. Lines 3-7 determine the association cost between the tracklets by using the Linear Cost Model defined in Section 3.6. Finally, Lines

Algorithm 3.1: Tracklet Association

Input: Video having F frames, A set of tracklets S^0
Output: A set of final tracklets S

1: Initialize θ_1^A, θ_2^A
2: **While** $\theta_1^A < \theta_{MAX}^A$ **Do**
3: **For** each tracklet T_j^i in S^i (a set of object trajectories) **Do**
4: **For** each tracket T_k^i in S^i which satisfies Eq. (3.1) **Do**
5: $A_{COST}(T_j^i, T_k^i) = \varphi(T_j^i, T_k^i)$
6: **End For**
7: **End For**
8: **For** each tracklet T_j^i in S^i **Do**
9: **If** there exists tracklet T_k^i which satisfies Eq. (3.2), (3.3), and (3.4) **Then**
10: Associate(T_j^i, T_k^i)
11: FillTemporalGap(T_j^i, T_k^i)
12: **End If**
13: **End For**
14: $\theta_1^A \leftarrow \theta_1^A + \theta_{STEP}^A$
15: **End While**
16: Return S (a set of final trajectories)

8-13 check the Eq. (3.2), (3.3) and (3.4), and associates the tracklets to form longer trajectories. The *FillTemporalGap()* function in Line 11 creates the set of detections U_{jk}^i which fills the object detections missing between the end of the Tracklet T_j^i and the start of the Tracklet T_k^i. The final result of the algorithm is a set of final object trajectories S. The association process is started with a lower threshold value θ_1^A, which allows only association of safe pairs, the tracklet pairs which have lower association cost. In every succeeding iteration, as solution space gets reduced and tracklets grow longer, the value of θ_1^A is increased by a step size θ_{STEP}^A, which allows more accurate association of difficult pair of tracklets. Tracklet association is continued until $\theta_1^A < \theta_{MAX}^A$.

3.5 Tracklet Creation

The Tracklet Creation step is used to form reliable tracklets by associating the detections obtained from a pre-trained detector. The reason for using the Tracklet Creation step is that initially, the given detection responses do not provide any motion information of the object. The independent object detections are associated only based on Euclidean distance. Detection d_i in frame f_i can be linked to at most one detection d_j from its preceding frame $f_i - 1$ and at most one detection d_k from its succeeding frame $f_i + 1$. The above two constraints considerably limit the search space of data association. Two detections $d_i^{f_i}, d_j^{f_j}$ can only be associated if they belong to two consecutive frames that is $| f_i - f_j | = 1$. Whether two detections $d_i^{f_i}, d_j^{f_j}$ could be associated is decided based on link cost $L(d_i^{f_i}, d_j^{f_j})$, which is given by the Euclidean distance between two detections.

$$L(d_i, d_j) = \sqrt{(x_j^{f_j} - x_i^{f_i})^2 + (y_j^{f_j} - y_i^{f_i})^2} \tag{3.6}$$

Similar to the dual threshold method in Tracklet Association, the object detections are linked to create short tracklets. Here, the term short tracklets is used because the strict threshold values are used to link the object detections because only the distance is used a cost. The idea is to link the object detections which are very close to each other, thus reducing the chances of erroneous links. Two object detections d_i (in any frame f) and d_j (in frame f+1) are linked if the link cost between these detections is:

(i) less than threshold θ_1^C:

$$L(d_i, d_j) < \theta_1^C \tag{3.7}$$

(ii) less than link cost of d_i and detection d_x in frame f+1, where θ_2^C is another threshold:

$$L(d_i, d_x) - L(d_i, d_j) > \theta_2^C, \ \forall \ d_x \ \in D^{f+1} - \{d_i, d_j\} \tag{3.8}$$

(iii) and less than link cost of $d_j$and detection d_x in frame f:

$$L(d_x, d_j) - L(d_i, d_j) > \theta_2^C, \ \forall \ d_x \ \in D^{f} - \{d_i, d_j\} \tag{3.9}$$

In Eq. (3.8) and (3.9), D_{f+1} is a set of object detections in frame f+1, and $D_f$is a set of object detections in frame f. Algorithm 3.2 shows the Tracklet Creation step of the GDA technique. In Lines 2-6, for every detection d_i in the set D_f, and for every detection d_j in the set D_{f+1}, the link cost between the detections d_i, d_j is calculated. Line 4 calculates the link cost between two detections. Based on the Eq. (3.7), (3.8), and (3.9), the object detections are linked iteratively by increasing the threshold value θ_1^C (superscript C refers to the Tracklet Creation step) in every iteration, as shown in Lines 8-15. The pairs of object

Algorithm 3.2: Tracklet Creation

Input: Video having F frames, set of object Detections D
Output: A set of tracklet S^0

1: **For** each frame f in F **Do**
2: **For** each detection d_i in D_f (set of Detections in frame f) **Do**
3: **For** each detection d_i in D_{f+1} (set of Detections in frame f + 1) **Do**
4: $\text{LinkCost}(d_i, d_j) = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}$
5: **End For**
6: **End For**
7: Initialize θ_1^C, θ_2^C
8: **While** $\theta_1^C \leq \theta_{MAX}^C$ **Do**
9: **For** each detection d_i in D_f **Do**
10: **If** there exists d_j in D_{f+1} which satisfies Eq. (3.7), (3.8), (3.9) **Then**
11: Link d_i with d_j
12: **End If**
13: **End For**
14: $\theta_1^C \leftarrow \theta_1^C + \theta_{STEP}^C$
15: **End While**
16: **End For**
17: Return S^0, where $S^o = \{T_k\}$ (a set of short tracklets)

detections with a lower link cost are associated before the remaining pairs. These object detections are continuously linked across the consecutive frames f, f+1 of the input video sequence, where F is the total number of frames. Line 17 returns the set of short tracklets, $S^0 = \{T_k\}$. The object detections with continuous associations form a single tracklet $T_k = \left\{ d_k^{f_k^s}, d_k^{f_k^{s+1}}, \dots, d_k^{f_k^e} \right\}$ and the end result of the Tracklet Creatsion step is a set of tracklets $S^0 = \{T_1, T_2, \dots, T_{L_0}\}$. Note that, even a single isolated detection, which does not have a link with other detections, is formed as a tracklet having length one.

3.6 Linear Motion Cost Model

Linear motion cost model similar to the motion model in [25] is used to calculate the association cost between tracklets T_i and T_j such that $0 < f_j^s - f_i^e < G_{max}$ where f_i^e is the last frame of T_i and f_j^s is the starting frame of T_j. The position of T_i is estimated at frame f_j^s, and position of T_j is estimated at frame f_i^e by using their respective velocities V_i, V_j and real positions:

$$P_j^{f_i^e} = P_j^{f_j^s} - V_j \times \Delta t \tag{3.10}$$

$$P_i^{f_j^s} = P_i^{f_i^e} + V_i \times \Delta t \tag{3.11}$$

where P_k^f is the position of tracklet T_k at frame f and $\Delta f = f_j^s - f_i^e$ is the temporal gap between the tracklets. The velocities V_i and V_j have two components: the slope component and the distance component. The slope is calculated using the position of the object in the last frame of the tracklet, and the position of the object in the (last-m) frame of the tracklet where m is some constant integer. The distance is calculated as an average distance travelled by the object during the corresponding m frames. Note that the velocity for the

preceding tracklet is calculated in the forward direction and the velocity for the succeeding tracklet is calculated in the backward direction. The association cost $\varphi(T_i, T_j)$ between T_i and T_j is proportional to the sum of Euclidean distances between the estimated positions of tracklets and their real positions. Here, dist_1 is the Euclidean distance between real position of T_j and estimated position of T_i at frame f_j^s. Similarly, dist_2 is the Euclidean distance between real position of T_i and estimated position of T_j at frame f_j^e.

$$\mathrm{dist}_1 = \mathrm{Distance}(P_j^{f_j^s}, P_i^{f_j^s}) \tag{3.12}$$

$$\mathrm{dist}_2 = \mathrm{Distance}(P_i^{f_i^e}, P_j^{f_i^e}) \tag{3.13}$$

$$\varphi(T_i, T_i) = \omega(\Delta f) \times (\mathrm{dist}_1 + \mathrm{dist}_2) \tag{3.14}$$

$$\omega(\Delta f) = \begin{cases} \log_{10}(\Delta f), & \text{if } \Delta f > 10 \\ 1, & \text{else} \end{cases} \tag{3.15}$$

Here, $\omega(\Delta f)$ is the FrameFactor and it is directly proportional to the number of frames. The use of frame factor encourages the association of two tracklets with a lesser temporal gap before the association of two tracklets with a higher temporal gap. Note that to keep the association cost within the limits of the threshold values, the maximum factor by which associated cost is increased (based on FrameFactor) is kept to 2. Since the value of frame gap (Δf) can vary from 0 to 100 in some cases (depending on G_{max}), we have used a log function to calculate the FrameFactor. The log function contracts the value to stay within the limit of 2 (for input of 0 to 100). As demonstrated by Fig. 3.3 (ii), based on the linear motion, the tracklet T_1 could directly associate with the tracklet T_3. In an ideal case, the tracklet T_1 should associate with the tracklet T_2 and the tracklet T_2 should associate with the tracklet T_3, as given in Fig. 3.3 (iii). Direct association of the tracklet T_1 with the tracklet

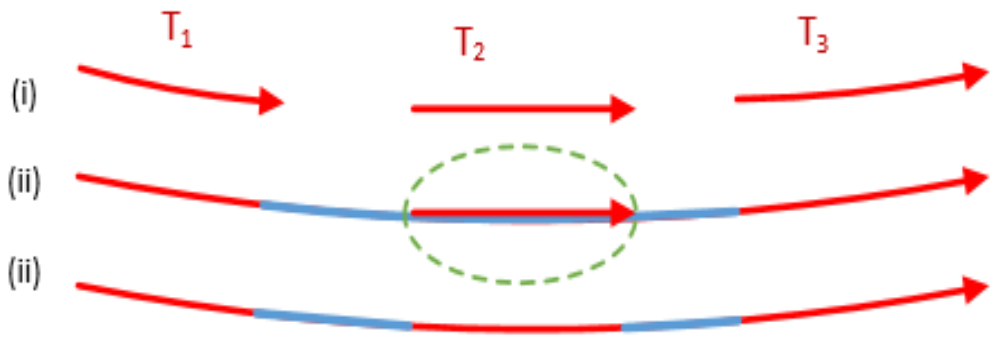

Figure 3.3: Illustrates the use of FrameFactor in GDA. Red line denotes actual tracklet, Blue line denotes positions generated by GDA, assuming detections are missing. Not using FrameFactor could increase False Positives by creating overlapping positions (shown by the green circle).

T_3 gives rise to false object detections (false positives) since GDA will generate estimated positions between the tracklets T_1 and T_3, as if the original detection responses were missing. However, use of FrameFactor increases the association cost between the tracklets T_1 and T_3, which enables the tracklet T_1 to first associate with the tracklet T_2, and then the tracklet T_2 to associate with the tracklet T_3.

3.7 Data Filtration

In the context of this chapter, data filtration refers to the process of removing the incorrect object detections in each frame. The detection hypothesis obtained from an object detector may not always be accurate and can have several incorrect object detections which do not exist in the *ground truth.* These incorrect object detections are termed as false positives and need to be found and removed. Two methods have been used for data filtration. The first method is based on confidence value, $conf_i$, associated with each of the given detections. This method is used in both the pre-data filtration and post-data filtration steps

of GDA (see Fig. 3.1). In pre-data filtration, a detection $d_i^{f_i}$ is considered to be a false positive if its confidence score $conf_i^{f_i}$ is less than α_1. Similarly, in post-data filtration, any tracklet T_k is removed if the average confidence score of all the detections $\left\{ d_k^{f_k^s}, d_k^{f_k^{s+1}}, \ldots, d_k^{f_k^e} \right\}$ is less than α_2. The second method of data filtration uses the object characteristics: width and height; and it is applicable only for video sequences captured with a stationary camera. From a stationary camera scene, it is possible to manually (by human vision) obtain the maximum and minimum height and width (maxH, minH, maxW, minW) of an object of interest in the scene. Based on these parameters, only the object detections which satisfy the two conditions $minH \leq h_i \leq maxH$ and $minW \leq w_i \leq maxW$ are considered to be true detections. Other object detections, which do not follow these conditions are removed as false alarms. Note that the second method of data filtration is used only in the pre-data filtration step.

3.8 Performance Analysis

This section discusses the performance analysis of the proposed GDA technique. The GDA technique is implemented in Python 2.7. All the experiments are performed on a system with 8 GB RAM and a 2.60 GHz dual-core i5 processor running the Ubuntu 14.04 LTS operating system.

3.8.1 Video Datasets

The two benchmark datasets, MOT16 [58] and PETS 2009 [59] are used for evaluation of the proposed GDA technique. MOT16 dataset has a total of 14 sequences: 7 sequences for training and 7 sequences for testing purpose. It includes challenges such as unconstrained environments and unseen sequences. As indicated in [58], for a fair comparison of tracking algorithms, MOT16 dataset provides detection hypothesis and *ground truth* annotations

carried out by well-qualified researchers. The *ground truth* consists of the object trajectories which are manually labeled by a human. They are deemed correct and forms the basis for determining the accuracy of a given MOT technique. The *ground truth* is available for the training sequences whereas access to the *ground truth* of testing sequences is restricted to prevent overfitting. Among the 7 testing sequences, 3 are captured with stationary camera and the remaining 4 by a moving camera. The various parameters, such as $\theta_1^A, \theta_2^A, \theta_{max}^A, \theta_1^C, \theta_2^C, \theta_{max}^C, G_{max}, \alpha_1, \alpha_2$,m used in the GDA technique are selected by using appropriate training sequence for each of the testing sequences. The values which produces best accuracy for training sequences are used. PETS2009 is one of the well-known datasets which was specifically created for evaluating video surveillance applications. From PETS2009 dataset, a PETS09-S2L1 sequence is used for comparison, in particular for the non-linear motion model based data association technique.

3.8.2 Performance Metrics

To study the performance of the proposed GDA technique, the qualitative performance metrics described in [58]. The under mentioned qualitative performance metrics were described in [58].

- **Number of False Positives (FPs)**: The occurrence of a target object in the output of a tracking algorithm when it is not available in the *ground truth* is considered a false positive and increments FPs by 1.
- **Number of False Negatives (FNs)**: If the target object is available in the *ground truth* but the tracking algorithm misses to locate it, then it is considered a false negative and FNs is incremented by 1.

- **Recall**: It is defined as the ratio of the sum of correctly matched objects (true positives) in the tracking results and the *ground truth* to the total number of objects in the *ground truth*. It is given by Eq. (3.16) where TPs is the number true positives.

$$Recall = \frac{TPs}{TPs+FNs} \tag{3.16}$$

- **Precision**: It is defined as the ration of the sum of correctly matched objects (true positives) in the tracking results and the *ground truth* divided to the total number of objects in the tracking results. It is given by Eq. (3.17) where TPs is the number of true positives.

$$Precision = \frac{TPs}{TPs+FPs} \tag{3.17}$$

- **Number of Identity Switches (IDS)**: IDS is the total number of identity switches in the object trajectories generated by a tracking algorithm in comparison to continuous object trajectories in the *ground truth*.
- **Frag**: The total number of discontinuities in the object trajectories of the tracking results as compared to the *ground truth*.
- **Mostly tracked targets (MT):** The ratio of ground-truth object trajectories that are covered by the object trajectories in tracking results for at least 80% of their respective life span.
- **Mostly lost targets (ML):** The ratio of ground-truth object trajectories that are covered by the object trajectories in tracking results for at most 20% of their respective life span.
- **Multiple Object Tracking Accuracy (MOTA)**: It is a measure that combines FPs, FNs, and IDS and is given by Eq. 3.18 where f is a frame index (a sequential ID of a

frame in a video sequence) and GT_f is a total number of objects in frame t of *ground truth* [58]. MOTA provides the overall accuracy of tracking algorithms.

$$\text{MOTA} = 1 - \frac{\sum_f(\text{FNs}_f+\text{FPs}_f+\text{IDS}_f)}{\sum_f \text{GT}_f} \tag{3.18}$$

- **Multiple Object Tracking Precision MOTP**: It gives the misalignment between the bounding boxes of the object trajectories in the *ground truth* and the bounding boxes of the object trajectories in tracking results. It gives the average overlap between the tracking results object trajectories and *ground truth* object trajectories. MOTP is given by Eq. (3.19) where nm_f denotes the number of matches in frame f and $bb_{f,i}$ is the bounding box overlap of target object i with its corresponding *ground truth* object.

$$\text{MOTP} = \frac{\sum_{f,i} \text{bb}_{f,i}}{\sum_f \text{nm}_f} \tag{3.19}$$

Processing speed is used as the quantitative measure for the performance of the GDA technique and is described below:

- **Hz**: Processing speed (in frames per second) of the achieved by running the algorithm on the benchmark. Note that the processing speed includes runtime of the data association stage only and not the object detection stage.

3.8.3 Results on MOT16 dataset

The average results of all 7 testing sequences of MOT16 is shown in Table 3.1. Note that in Table 3.1 and the subsequent tables (3.2 to 3.8) an upward arrow means the higher the value is, the better is the result and a downward arrow means the lower the value is, the better is the result. As mentioned, the major focus of the proposed GDA technique is to generate quick results. As shown in Table 3.1, the GDA technique produces computationally faster

results with the processing speed of 182.7 frames per second, whereas other top four methods JMC [11], MHT [50], TBD [44], CEM [12] range only between 0.3 to 2.6 frames per second. The overall qualitative performance, given by MOTA, of GDA on the 7 testing sequences is 43.3% and is comparable to the existing state-of-the-art methods. Also, the GDA technique has the second lowest FNs which shows the effectiveness of creating the missing object detections between the object trajectories. In the results presented in Table 3.1, GDA has the highest FPs. The major source for this higher FPs is the inaccuracies in the object detector. Although the data filtration is used to remove the false positives, to achieve a lower FPs by using data filtration also leads to a higher FNs. The trade-off between FPs and FNs can be seen from the results in Table 3.1. The GDA technique has the highest FPs, and also has the second lowest FNs. Comparing with the state-of-the-art methods, globally-optimal greedy algorithms-based tracking method DPNMS [45] has the lowest FPs with the highest FNs. To lower FPs, DPNMS uses greedy algorithm for non-maxima

Table 3.1: Comparison of results of MOT16 dataset averaged over all 7 testing sequences. Results of other methods taken from the official website of MOT16 dataset [21]. Total number of frames: 5919

Method	MOTA ↑	MOTP ↑	MT ↑	ML ↓	FPs ↓	FNs ↓	IDS ↓	Frag ↓	Hz ↑
JMC	46.3	75.7	15.5	39.7	6,373	90,914	657	1,114	0.8
GDA	**43.3**	**74.3**	**11.9**	**42.8**	**8,463**	**93,892**	**985**	**1,509**	**182.7**
MHT	42.9	76.6	12.6	46.9	5,668	97,919	499	659	0.8
TBD	33.7	76.5	7.2	54.2	5,804	112,587	2418	2252	1.3
CEM	33.2	75.8	7.8	54.4	6,837	114,322	642	731	0.3
DPNMS	32.2	76.4	5.4	62.1	1,123	121,579	972	944	212.6

suppression of the input object detections. In the process, DPNMS leads to a significantly higher FPs, approximately 29% higher than GDA. Although a higher FPs is a disadvantage, producing a lower FNs at the cost of higher FPs may be more helpful for surveillance analysis while searching for suspected threats. In terms of computational speed, DPNMS is relatively faster than GDA, however, GDA has 11% higher accuracy (MOTA) than DP_NMS.

As shown in Table 3.1, the GDA technique produces a relatively higher IDS and Frags as compared to other state-of-the-art methods (JMC, MHT, CEM, DPNMS) whereas highest IDS are produced by TBD. A higher IDS is possibly the drawback of not using an appearance model for calculating the association cost between the object trajectories. Despite having higher IDS, it is important to note that the GDA technique is not restrictive and can be used in applications where appearance information is either not available or cannot be easily extracted, or several objects may have similar appearances. More importantly, the GDA technique is 50 to 600 times faster in comparison to JMC, MHT, TBD and CEM.

The results shown in Table 3.1 are the average results of the 7 testing sequences of the MOT16 dataset. For each of the seven testing sequences: MOT16-01, MOT16-03, MOT16-08, MOT16-06, MOT16-07, MOT16-014, and MOT16-14; the tracking results are shown in Table 3.2, Table 3.3, Table 3.4, Table 3.5, Table 3.6, Table 3.7 and Table 3.8 respectively. The GDA technique has comparable qualitative performance to the state-of-the-art methods on each of the sequences. The few points that are not captured by the average results are discussed in the next paragraph.

The sequences MOT16-01, MOT16-03 and MOT16-08 are captured with a stationary camera whereas the rest of the four sequences are captured with a moving camera. As shown in the results, the proposed GDA technique has relatively better performance on the video sequences that are captured with a stationary camera when compared to the video sequences which are obtained using a moving camera. When a video is captured with a moving camera, the objects are moving as well as the background is continuously changing, thus resulting in a relatively faster object motions. Also, the moving camera introduces non-linearity in the motion of every object. Since the GDA technique is based only on the linear motion model, the qualitative performance of GDA on the sequences with a moving camera is relatively lower.

Table 3.2: Comparison of results of MOT16-01 sequence. Number of frames: 450. Average crowd density: 14.2. Captured with a stationary camera

Method	**MOTA** ↑	**MOTP** ↑	**MT** ↑	**ML** ↓	**FPs** ↓	**FNs** ↓	**IDS** ↓	**Frag** ↓
GDA	**31.4**	**71.2**	**17.4**	**47.8**	**140**	**4211**	**39**	**63**
JMC	30.3	72.4	13	43.5	132	4287	40	54
MHT	30.1	72.8	26.1	47.8	164	4294	15	30
CEM	28.4	72.3	17.4	43.5	123	4420	36	31
TBD	22.4	73.7	8.7	56.5	82	4814	65	74
DPNMS	20.2	73.9	4.3	65.2	19	5051	31	38

Table 3.3: Comparison of results of MOT16-03 sequence. Number of frames: 1500. Average crowd density: 69.7. Captured with a stationary camera.

Method	MOTA ↑	MOTP ↑	MT ↑	ML ↓	FPs ↓	FNs ↓	IDS ↓	Frag ↓
JMC	53.5	75.9	23	19.6	3,144	45,223	264	422
GDA	53.0	74.4	19.6	19.6	3,653	45,266	186	570
MHT	49.0	76.5	18.9	27.0	3,591	49,521	230	304
TBD	39.5	76.5	8.8	29.7	3,435	58,280	1,509	1,438
CEM	38.3	75.9	11.5	39.9	3,251	61,110	168	238
DPNMS	37.7	76.3	6.1	40.5	809	63,873	490	534

Table 3.4: Comparison of results of MOT16-08 sequence. Number of frames: 625. Average crowd density: 26.8. Captured with a stationary camera

Method	MOTA ↑	MOTP ↑	MT ↑	ML ↓	FPs ↓	FNs ↓	IDS ↓	Frag ↓
JMC	32.9	79.9	11.1	36.5	393	10645	82	92
MHT	32.4	80.9	12.7	44.4	331	10903	76	77
GDA	**29.8**	**77.5**	**12.7**	**41.3**	**1030**	**10602**	**115**	**128**
DPNMS	27.5	80.4	9.5	52.4	100	11958	77	70
CEM	26.7	80	9.5	47.6	632	11553	77	74
TBD	25.9	80.7	11.1	46	749	11477	170	135

Table 3.5: Comparison of results of MOT16-06 sequence. Number of frames: 1194. Average crowd density: 9.7. Captured with a moving camera

Method	MOTA ↑	MOTP ↑	MT ↑	ML ↓	FPs ↓	FNs ↓	IDS ↓	Frag ↓
JMC	49.6	73.1	16.3	41.6	375	5346	97	183
MHT	46.7	75.1	15.8	53.4	247	5840	62	84
GDA	**45.5**	**72.6**	**16.3**	**42.5**	**555**	**5554**	**191**	**212**
TBD	38.5	74.7	8.6	57	209	6692	196	142
DPNMS	34.6	75.1	5.4	63.3	40	7326	182	102
CEM	33.9	74.2	6.8	56.6	527	6960	138	143

Table 3.6: Comparison of results of MOT16-07 sequence. Number of frames: 500. Average crowd density: 32.6. Captured with a moving camera

Method	MOTA ↑	MOTP ↑	MT ↑	ML ↓	FPs ↓	FNs ↓	IDS ↓	Frag ↓
JMC	42.8	74	11.1	27.8	914	8346	74	144
GDA	**39.9**	**73.1**	**11.1**	**29.6**	**632**	**9063**	**109**	**214**
MHT	36.6	75.8	11.1	42.6	408	9896	39	60
CEM	28.3	74.1	7.4	48.1	793	10808	107	126
TBD	27.5	74.8	7.4	50	682	10938	217	1213
DPNMS	27.2	75	7.4	61.1	48	11747	86	93

Table 3.7: Comparison of results of MOT16-14 sequence. Number of frames: 750. Average crowd density: 24.6. Captured with a moving camera

Method	MOTA ↑	MOTP ↑	MT ↑	ML ↓	FPs ↓	FNs ↓	IDS ↓	Frag ↓
JMC	26.2	73.8	7.9	56.7	715	12855	70	177
MHT	25.8	75.6	3.7	56.7	661	12985	62	77
GDA	**21.5**	**73.5**	**1.2**	**62.2**	**593**	**13636**	**277**	**200**
CEM	15.1	73.4	1.2	72	860	14752	82	84
TBD	13.5	75.9	1.2	77.4	175	15632	185	185
DPNMS	9.3	75.3	0.6	86.0	40	16642	73	73

Table 3.8: Comparison of results of MOT16-12 sequence. Number of frames: 900. Average crowd density: 9.2. Captured with a moving camera

Method	MOTA ↑	MOTP ↑	MT ↑	ML ↓	FPs ↓	FNs ↓	IDS ↓	Frag ↓
MHT	42.6	78.5	16.3	50.0	266	4480	15	27
JMC	41.6	77.6	22.1	45.3	700	4112	30	42
DPNMS	38.7	77.3	9.3	57	67	4982	33	34
TBD	36.1	77.8	9.3	52.2	772	4754	76	65
CEM	34.9	77.9	12.8	52.3	651	4719	34	35
GDA	**9.6**	**73.1**	**5.8**	**54.7**	**1860**	**5560**	**78**	**122**

As shown in Table 3.8, results for MOT16-12 sequence, the GDA technique has the lowest MOTA value. As the video is captured by a camera which is installed on a moving bus, GDA does not have a good performance. The speed of the bus is relatively faster than a human and the turns taken by the bus introduce a significant non-linearity in the objects' motion. Although MOT16-14, MOT16-06 and MOT16-07 are also captured with a moving camera, the camera is moving at a normal walking speed of a human. Therefore, despite the use of moving camera for video capture, the GDA technique has comparable tracking accuracy (MOTA) with other state-of-the-art methods for these video sequences. Overall, the GDA technique performs relatively better on the video sequences captured with a stationary camera.

3.8.4 Iterative matching vs non-iterative matching (MOT16 dataset)

The GDA technique uses several iterations where the threshold is increased in every iteration to decrease the strictness of matching. The threshold-based technique is also used in [48], where a fixed threshold value is used in a single iteration. To show the effectiveness of the iterative matching by varying the threshold, comparison of fixed threshold with varying threshold (at the Tracklet Association Step) is given in Fig. 3.4, for the training sequence MOT16-09. Note that here a training sequence is used for comparing the results, since results are evaluated multiple times by varying the θ_{max}^{A}. For testing sequences, the *ground truth* is not available, and results can be submitted online [60] for evaluation only once in a period of 3 days to prevent the overfitting on the testing sequences. In Fig 3.4, grey color label represents MOTA value for fixed threshold θ_{max}^{A}, whereas black color

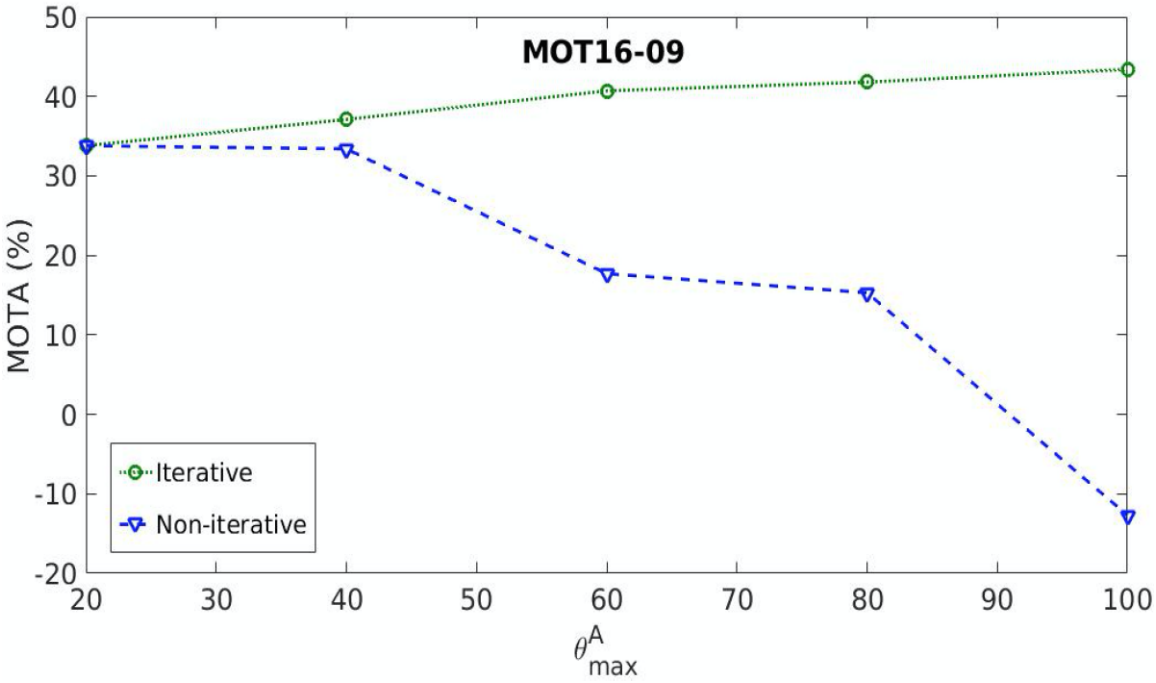

Figure 3.4: **Performance of MOT16-09 sequence with and without iterative method.**

label represents MOTA value for iteratively increasing threshold starting from $\theta_1^A = 10$ till θ_{max}^A, increasing by step size θ_{STEP}^A=10 in each iteration. For the iterative threshold-based method, larger the value of θ_{max}^A, higher is the accuracy. Starting from a lower threshold value θ_1^A, the pair of tracklets with lower association cost get associated. In further iterations, solution space is reduced and tracklets grow longer which give more accurate object motion information, and this reduces the chances of erroneous associations. For a fixed threshold-based method, accuracy (MOTA) decreases with increase in θ_{max}^A. This is expected since larger the value of θ_{max}, higher are the chances of erroneous associations if associations are completed in a single iteration. For a fixed value of θ_{max}^A, iterative method always performs better than fixed threshold-based method. For instance, at $\theta_{max}^A = 100$, MOTA of iterative method is 43.4% as compared to -12.7% of fixed threshold method. Also note that in Fig. 3.5, as the value of θ_{max}^A is increased in multiple

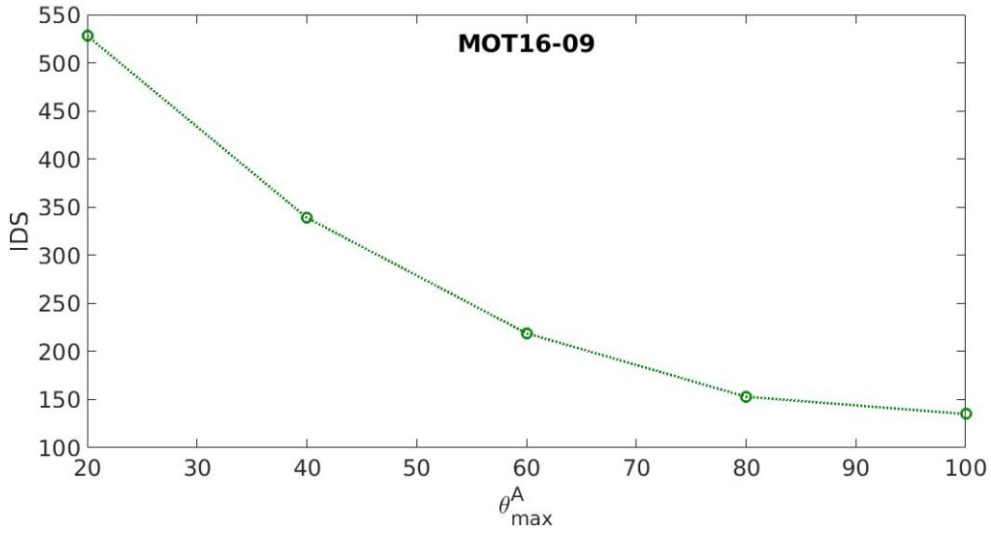

Figure 3.5: **Comparison of Identity Swaps for given θ^{A}_{max}.**

iterations, the number of IDS reduces. This shows that with every iteration, more and more trajectories are associated correctly.

3.8.5 Results on PETS09-S2L1 sequence

The performance evaluation on the MOT16 dataset shows the effectiveness of the proposed GDA technique, especially on the sequences with a stationary camera. For a moving camera sequence, particularly for a fast-moving camera, the GDA technique does not produce comparative accuracy with the state-of-the-art methods. As all the objects in the sequence have significant non-linearity in their motions. Although the GDA technique uses a linear motion model, it can still partly handle non-linearity in object motions, when a video sequence is captured with a stationary camera. The non-linear object motion in these cases are possibly approximated by piece-wise linear motion models. Note that the non-linearity in the video sequence can be observed by visual inspection. The PETS2009-S2L1 sequence has a mixture of linear and non-linear object motion (see Fig. 3.6). In Table 3.9, the results of the GDA technique for the PETS2009-S2L1 sequence are compared with two

other methods: MOT by online learning of non-linear motion patterns and robust appearance models (NLM) [13], and tracking by person identity recognition (PRIMPT)for MOT [10]. Here, [13] uses a non-linear motion model and the method proposed in [10] uses online learned discriminative appearance models to identify each object. Although the GDA technique which only uses a linear motion model, it still has outperformed these state-of-the-art methods in terms of a recall value. Comparing with results of [13] and [10], GDA has 5% and 7% higher recall value respectively. Despite not using any appearance model, GDA is partly able to handle non-linear object motion and large frame gaps. This is demonstrated by higher MT value, 94.73%, of GDA method, which is approximately 5% and 16% more than that of [13] and [10, 5] respectively. The rationale for comparable performance with the state-of-the-art methods, which use non-linear motion patterns, and discriminative appearance models, is based on two aspects. Firstly, the Tracklet Creation step only uses Euclidean distance as a cost model which associates the safe pairs of detections even with non-linear motion, but this works only when the frame gap is zero. Secondly, with every iteration in the Tracklet Association step, the size of solution space gets smaller, and the strictness of matching is reduced. This allows the association of the

Table 3.9: Comparison of results of PETS2009-S2L1 sequence. Results of NLM and PRIMPT taken from [13, 22]**. Total number of frames: 795**

Method	Recall ↑	Precision ↑	MT ↑	ML ↓	Frag ↓	IDS ↓	Hz ↑
GDA	96.94	98.09	94.73	0.0	22	8	345
NLM	91.8	99.0	89.5	0.0	9	0	16
PRIMPT	89.5	99.6	78.9	0.0	23	1	-

difficult pair of object trajectories even when the trajectory of object changes non-linearly. The non-linear motion handling of GDA for the sequence with a stationary camera can also be seen in visual results of the GDA technique for the PETS09-S2L1 sequence, shown in Fig. 3.6. In frame 275, object 2, 7, 16 and 17 are moving towards each other. In frame 296, objects 2 and 16 are overlapping. In frame 316, the object 2 after taking a non-linear turn is still tracked with a same id. Also in frame 335, the objects 16 and 17 after taking a U-turn, are still tracked with a correct id. It shows that the GDA technique is partly able to handle a non-linear object motion, if a video is captured with a stationary camera.

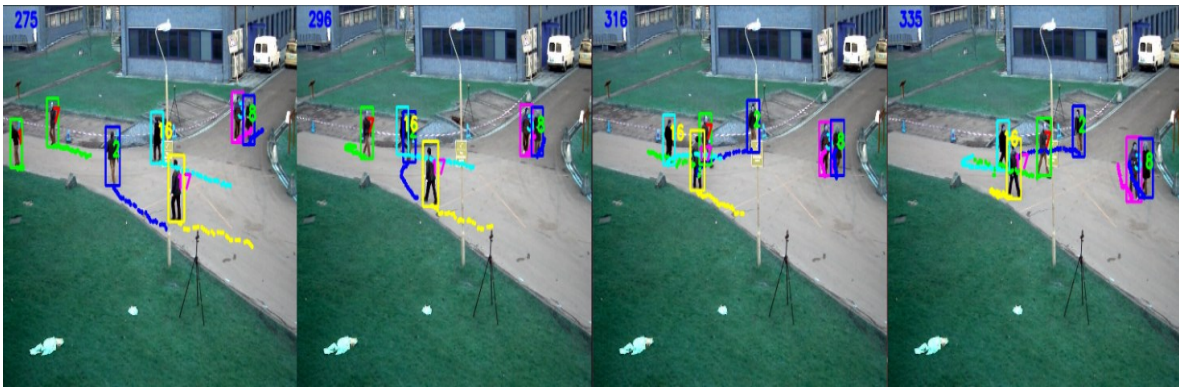

Figure 3.6: Results of PETS2009-S2L1 sequence. GDA is able to handle object occlusions and object interaction. Figure shown best in color.

3.9 Summary

This chapter presented a MOT method which incorporates a greedy data association technique (GDA) and mainly focuses on obtaining faster processing speed. The performance analysis of GDA is done using the benchmark video datasets. By using an iterative algorithm, the GDA technique shows that without using any appearance model and global optimization techniques, it is possible to achieve lower runtime while maintaining almost the same qualitative performance achieved by the current state-of-the-art methods. GDA technique is able to handle some non-linearity in the motion of objects

but the performance deteriorates when there is a high level of non-linearity in the video frames. Computationally fast methods, like GDA, are beneficial for large-scale video surveillance analysis, when the average size of recorded video footages is in Terabytes.

Chapter: 4 MapReduce-based Techniques for MOT

This chapter discusses performing MOT using MapReduce-based techniques. There are two parts of this chapter. The first part introduces three Map-Reduce based techniques (MOT-MR) for parallelizing MOT on a Hadoop cluster and the second part discusses the performance analysis of the MapReduce techniques. The MOT-MR techniques presented in the first part of the chapter are agnostic of the object detection and data association methodologies. As a result, the proposed MOT-MOR techniques can be adapted to any detection and data association algorithm found in the literature. The second part of the chapter uses specific algorithms including the proposed GDA to analyze the performance analysis of the MOT-MR techniques.

4.1 MOT-MR: MapReduce-based techniques for parallelizing MOT.

Apache Hadoop, discussed in Section 2.2.1, provides a useful platform for programs analyzing big datasets that include video analytics applications. As discussed in Section 2.2.1, it comprises of a single master node and multiple slave nodes. For processing a video file (V) on Hadoop, the video file is split into a set CS={c_i} of multiple video chunks and these chunks are stored on HDFS. If F is the total number of frames in V, then video chunk c_i will have f_i frames where $f_i \cong (F/C)$ and C denotes the cardinality of the set CS. Each map task operates on one of the chunks independently. The intermediate outputs from these map tasks are combined by the reduce task. Note that the outputs of map tasks are continuous for each data chunk but discontinuous for the video file V (the object trajectories will not remain continuous if an object appears in more than one chunk). These discontinuous outputs are combined by a single reduce task to generate the final object trajectories which are continuous and correspond to the original video file V. Also, note that higher the number of

chunks a video file is split into, more is the overhead of combining the intermediate results of these chunks in the reduce task. It is assumed that initially, the video file V is available on the master node of the Hadoop cluster. Before uploading the file to the HDFS, it is split into C chunks by using the FFMPEG tool [61]. The value of C is equal to the number of slave nodes (N) in the Hadoop cluster so that all the cluster nodes are utilized and reduce task overhead of combining the results is also minimized. Each video chunk c_i is given a unique and sequential id i. Since a list of the outputs received by the reduce task from map task is not ordered, this unique identity, i, is used to combine the results by maintaining the original sequence of the video V.

As discussed in Section 1.2, the object detection stage of MOT is time-independent, so splitting a video file and processing the multiple video chunks on a multi-node cluster does not have any impact on the qualitative performance (accuracy) of the object detection stage. However, the data association stage of MOT is time-dependent. Splitting a video file and executing the data association method on the multiple video chunks independently on a multi-node cluster can lead to variations in the tracking accuracy when compared with the tracking accuracy achieved on a single node. To deal with the time-dependencies, three different techniques are presented for implementing MOT on a MapReduce platform in the following sections.

4.1.1 Partially Parallel Technique (PP)

The partially parallel technique separates the detection stage and tracking stage of MOT in the Map phase and the Reduce phase respectively. As shown in Fig. 4.1, the basic idea is to execute only the time-independent processes (the object detection stage) in parallel (in Map phase) and execute the time-dependent processes (the data association stage)

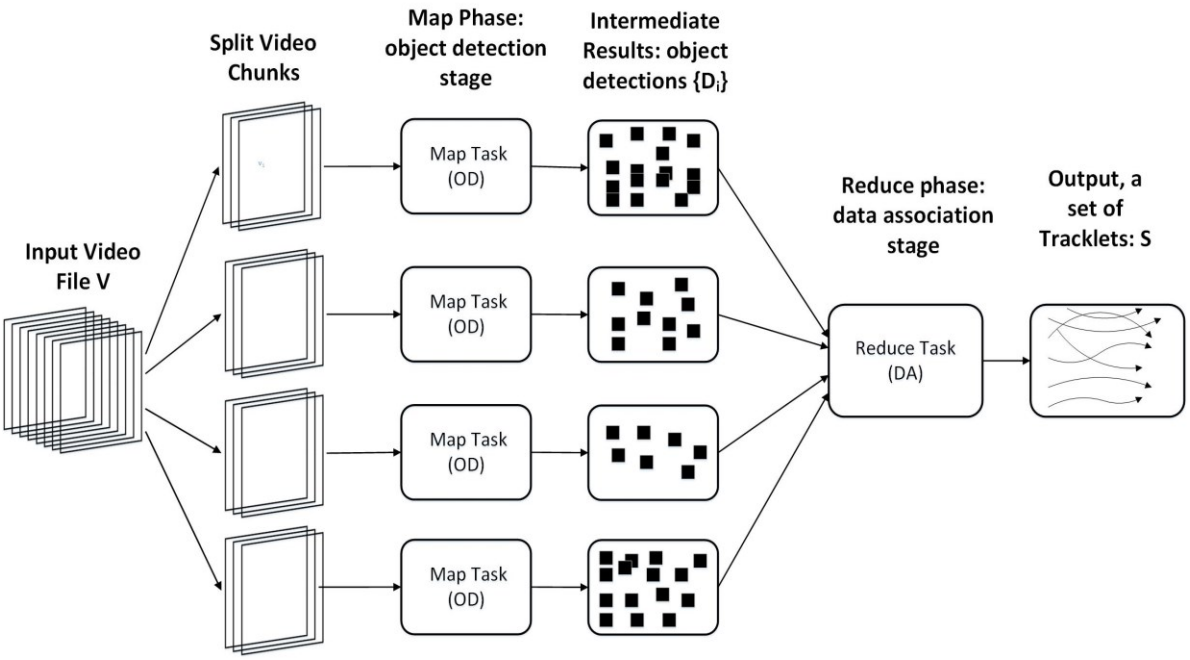

Figure 4.1: An Overview of the Partially Parallel Technique.

sequentially (in Reduce phase). Algorithm 4.1 illustrates the steps in the PP technique. Initially, the video file V is split into a set of C video chunks CS={c_i} as given in Line 1, where C=N. Lines 3-6 show the Map phase and Lines 7-10 show the Reduce phase. During the Map phase, the input for each of the map task is a key-value pair <i,c_i>, where key i is the unique identity of a video chunk and value c_i is a video chunk file (Line 4). A map task produces a set of object detections D_i, by invoking the object detection method, Detector.getDetections(c_i), as an intermediate output for the video chunk c_i. The output from each map task is a key-value pair <ς,D_i> where ς is a constant with an arbitrary value (Line 5). Note that the shuffle and sort phase of MapReduce transfer the intermediate outputs based on the key in the key-value pair. The intermediate outputs which have the same key are transferred to the same reduce task. So, the same constant ς is used as a key for all the intermediate map task outputs, so that these outputs are combined in the single reduce task. Combining the outputs in a single node is important because the final output

Algorithm 4.1: The Partially Parallel technique

Input: a video sequence V, number of nodes N
Output: a set of object trajectories, S

```
1:  CS = splitVideo(V, N)
2:  Upload CS to HDFS and start Hadoop Job
3:  MapPhase (< i, ci >):
4:      Di = Detector.getDetections(ci)
5:      return (ς,Di)
6:  End MapPhase
7:  ReducePhase (<ς,{Di)>):
8:      D=sortAndAppend({Di})
9:      S=Tracker.getTracklets(D)
10:     return ('Trajectories',S)
11: End ReducePhase
12: Copy S to local-file-system of the Master node
```

should be a set of object trajectories which are continuous for the big video file. The reduce task on obtaining the list of object detections $\{D_i\}$ sorts and appends all the detection responses into one set $D=\{D_1,D_2, \ldots, D_N\}$ by calling sortAndAppend($\{D_i\}$) (Line 8) and executes the data association method, Tracker.getTracklets(D), to obtain S, the set of final object trajectories (Line 9). Note that the list is sorted in an increasing order of unique identity i (which was assigned to each video chunk) of each detection so that the correct sequence of video V is maintained. The reduce task produces a set of final object trajectories S (Line 9). After the Hadoop program ends, S is copied to the local file system of the master node. One of the advantages of PP is that it provides no loss in the qualitative performance of MOT when compared with performance on a single node because the data association stage is executed sequentially by a single reduce task (on a single node). However, the sequential implementation of the data association stage is also a bottleneck for the speedup. The faster the data association method, the lesser is its effect on the speedup achieved by using multiple processing nodes. Thus, this technique is more useful

for greedy data association methods such as [15, 45] which are computationally fast.

4.1.2 Fully Parallel Technique (FP)

In the fully parallel technique, the main idea is to parallelize both the time-independent and time-dependent processes. As shown in Fig. 4.2, each map task produces the object trajectories for a single video chunk and these object trajectories are combined by a single reduce task. Algorithm 4.2 illustrates the steps in the FP technique. The Map phase implements the full MOT method (the object detection stage and data association stage) (Lines 3-7). As in the case of PP, input for each the map task is a key-value pair <i, c_i>. Each chunk c_i of the video is processed independently on a single map task. A set of object detections, D_i, is obtained by invoking the method *Detector.getDetections(c_i)* (Line 4) and the data association is performed on these object detections by invoking the method

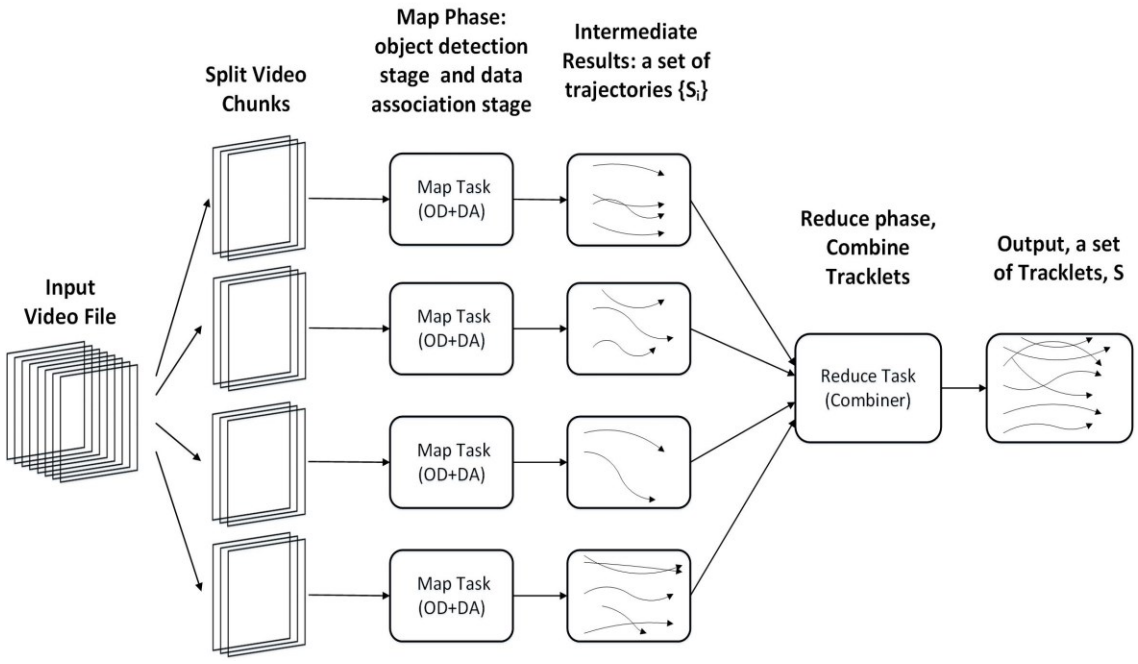

Figure 4.2: An overview of the Fully Parallel Technique.

Algorithm 4.2: The Fully Parallel Technique.

Input: a video sequence V, number of node N
Output: a set of object trajectories, S

```
1:  CS = splitVideo(V,N)
2:  Upload CS to HDFS and start Hadoop Job
3:  MapPhase (< i, ci >):
4:      Di = Detector.getDetections(ci)
5:      Si = Tracker.getTracklets(Di)
6:      return(ς,Si)
7:  End MapPhase
8:  ReducePhase (<ς,{Si} >):
9:     sort({Si})
10:    S = S1
11:    For i = 2 to N Do:
12:        S = Tracker.combine(S, Si)
13:    End For
14:    return ('Tracklets',S)
15: End ReducePhase
16: copy S to local-File-System of the Master node
```

Tracker.getTracklets(D_i) (Line 5). The result of each map task is a key-value pair $< \varsigma, S_i >$ where ς is some constant and S_i is a set of tracklets obtained from video chunk c_i. Similar to PP, the constant ς is used as a key for all the intermediate outputs, so that the final results are combined into a single set of trajectories which are continuous for video V. The reduce task calls *sort{S_i}* and sorts the tracking results obtained from all the map tasks in increasing order of i to maintain the sequence of video V (Line 9). For combining N set of tracklets, N-1 operations are used by calling *Tracker.combine(S, S_i)*, where each operation combines and appends the set S_i to the set S (Line 12). Here, S is a set obtained by combining previous i-1 sets. Note that for implementing MOT on a single node, usually two methods: the object detection method and the data association method are implemented. In addition to these two methods, a *Tracker.combine()* method is required to be implemented for MOT-MR techniques. Although the implementation of *Tracker.combine()* solely depends on the data association method used, the purpose of *Tracker.combine()* is to join the end of trajectories

of chunk c_i to the start of the respective trajectories of chunk c_{i+1}. In doing so, *Tracker.combine()* processes very few frames in comparison to the data association method. The independent tracking results from all chunks are combined in the reduce task, depending on the data association method, so the qualitative performance of FP on a multi-node cluster can be different when compared to the qualitative performance obtained on a single node. The reason is that instead of using the global optimization techniques over the complete video file, the optimization techniques are applied for each video chunk independently in which case the affinity measures from adjacent chunks are not available. For instance, during sequential processing, the data association stage for video V uses affinity measures such as object motion and object appearance from preceding and succeeding frames. However, when the video V is split, each independent chunk c_i will not have access to the frames from adjacent chunks and as a result object motion and object appearance from these inaccessible frames are not used. This can lead to a difference in qualitative performance when a video is split into multiple of chunks.

4.1.3 Fully Parallel Technique with Frame Overlap (FPO)

The lack of affinity measures (discussed in Section 2.1.1) from adjacent frames at the start and end of each video chunk can be handled up to some extent by splitting the video file with overlapping frames: a fully parallel technique with frame overlap (FPO). Note that the distribution of time-dependent and time-independent processes in FPO is similar to FP as shown in Algorithm 4.2. The difference between FP and FPO is the splitting of the video file V. In FPO, after splitting a video file V, each chunk c_i (for i=2 to N) will have a $\delta+f_i$ number of frames. Here, δ is the number of frames which are overlapped between chunks. For example, if 1000 frames video is split into two chunks (with $\delta = 30$), the chunks will

have 1-520 frames and 491-1000 frames respectively (one possibility). The first δ frames of each video chunk c_i (i=2 to N) are the copy of last δ frames of chunk c_{i-1}. The overlapping of frames provides motion and appearance affinity measures which were missing from the adjacent video chunks. With overlapping, each video chunk c_i (i=2 to N) will have δ additional frames to be processed in the map task, which is an overhead. In the reduce task, *Tracker.combine()* removes the additional trajectories produced due to overlapping frames and produces the final set S of object trajectories which correspond to the original video V. In the performance analysis presented in Section 4.2.4, it is shown that the use of frame overlapping in FPO can enhance qualitative performance compared to FP. FP and FPO are useful when computationally slow data association algorithms are used because both of these techniques parallelize operations in both the object detection and the data association.

4.1.4 Video Splitting

The video file to be analyzed for MOT is required to be split into multiple chunks for using the MapReduce platform. For an ideal case, the size of each video chunk (number of frames) should be equal, so that all map tasks finish executing at an approximately the same time. For example, a video file having 1050 frames will be split into 6 chunks each having 175 frames. The reduce task will not start until all the map tasks are finished. If video chunks have an unequal number of frames, then map tasks will also have unequal execution times. This will lead to inefficient use of the cluster nodes since some of the nodes will become idle by finishing smaller video chunks, while others will take longer to finish the larger chunks.

Video splitting is executed sequentially on the master node and it is an overhead. For splitting a video into an exactly equal number of frames per chunk, the video is required to be re-encoded which is computationally very expensive. A computationally fast video splitting algorithm (FFMPEG segments-based method) [61] splits the video based on the key-frame which does not require video re-encoding. However, this method splits the video file into multiple chunks where each chunk will have a number of frames that is a multiple of a *group of pictures (GOP)*, a parameter used while video encoding [61]. In video encoding, GOP is a collection of successive pictures which specify the order in which the key-frames are arranged. A video file does not contain complete information of each frame but contains sets of grouped frames that refer to each other to produce a set of complete frames [62]. For example, if the GOP value is 10 (fixed), the video file having 1050 frames will be encoded into sets of grouped frames where each group will have 10 frames. When splitting this video file into multiple chunks by using FFMPEG segment-based method, the sets of grouped frames are copied to the resulting split chunks. As a result, each chunk will have a number of frames which are a multiple of GOP. For example, splitting a 1050 frames video file will generate 5 chunks having 180 frames and 1 chunk having 150 frames (instead of each chunk having 175 frames when re-encoding used). The output chunks may not have an equal number of frames, but the overhead of video splitting is reduced by a large factor when the GOP-based splitting method is used [61].

4.2 Performance Analysis of the MOT-MR Technique

In this section, the performance analysis of the proposed MOT-MR techniques is given. The qualitative performance and the computational speedup for the MOT methods on a Hadoop cluster are compared with performance on a single node.

4.2.1 Implementation details

The evaluation of MOT-MR methods is based on prototyping and measurement. Apache Hadoop and HDFS are used for implementation of the MOT-MR based techniques. Amazon web services based elastic cloud compute (EC2) is used for setting up a Hadoop cluster. Each node of the Hadoop cluster is a compute-optimized C4.XLARGE instance on EC2 (having 2.9 GHz CPU clock frequency and 7.5GB RAM) [56]. Note that one of the CPU instances used in Hadoop cluster hosts both slave node and master node whereas each of the remaining instances hosts a single slave node. A maximum of 20 nodes is used on the AWS EC2 cloud to limit the cost that accrues from the acquisition of nodes. For the object detection stage, the existing state-of-the-art the Deformable Part-based Models (DPM) object detector presented in [7, 32] is used as the algorithm is readily available and is used by a number of researchers [11, 58, 45, 47, 20, 23]. For the data association, the GDA technique proposed in this thesis is used. It is important to note that in the performance analysis of MOT-MR techniques, the focus is on analyzing the relative performance of the different MOT-MR techniques and the specific DPM and GDA algorithms that are common to each technique are not important.

4.2.2 Video Dataset

One video sequence each from 2DMOT2015 [63] and MOT6 [58] benchmark datasets is used for the evaluation of the MOT-MR techniques. These datasets include the *ground truth* and the programs used for measuring the qualitative performance of tracking results. The *ground truth* consists of the object trajectories which are manually labeled by a human. They are deemed correct and forms the basis for determining the accuracy of a given MOT technique. From 2DMOT2015, the PETS09-S2L1 (PETS) sequence is used which has an

average crowd density of 5.6 per frame whereas from MOT16, the MOT16-04 (MOT4) sequence is used which has an average crowd density of 45.3 per frame. An average crowd density of the video sequence is the average number of objects present in each frame. The datasets MOT16 and 2DMOT2015 are available as a sequence of images, thus, the FFMPEG tool [61] is used to create videos from these image sequences. The PETS sequence has 795 frames whereas the MOT4 sequence has 1050 frames. Although these datasets are not very large, slow MOT methods are used to demonstrate the effectiveness of the proposed MOT-MR techniques; thus, these datasets are appropriate for investigating the effectiveness of parallel processing on the Hadoop. In addition to these two video sequences, an extended video file (PETS-E) by using the same PETS09-S2L2 sequence is created. The same video sequence is repeated back and forth alternatively (1 to 795 frames followed by 795 to 1 frames, then followed by 1 to795 frames and so on) so as to obtain a large video file which has continuous trajectories of moving objects. The resulting PETS-E sequence has 15000 frames.

4.2.3 Experimental Setup

The performance analysis of the proposed MOT-MR techniques is evaluated by comparing the qualitative performance and the system performance. The various parameters used for the experiments are discussed next.

4.2.3.1 Number of Nodes (N)

The experiments are conducted on a single node and on 2, 4, 6, 8, 10 nodes in a Hadoop cluster (for the small video sequences PETS and MOT4) and on a single node and 5, 15, 20 nodes (for the large video sequence PETS-E). A larger number of nodes is used in

experiments with the larger sequence (PETS-E) because of the larger computational requirement associated with the processing of larger the video files.

4.2.3.2 δ (Number of overlap frames in FPO)

A value of δ=10 is used for the PETS and PETS-E sequences and a value of δ=40 is used for the MOT4 sequence. A higher value of δ is chosen for the MOT4 sequence because it has a higher crowd density. A higher crowd density video has a higher number of object trajectories. Using a larger δ value helps in differentiating the object trajectories from one another by providing more accurate affinity measures from adjacent video chunks.

4.2.3.3 GDA technique for data association

The data association technique, GDA, used for the experiments is a fast greedy method (345 frames per second for the PETS sequence) that was described in Chapter 3. The qualitative performance measures discussed in Section 3.8.2 of Chapter 3 shows that qualitative performance of the GDA technique is comparable to the qualitative performance of state-of-the-art algorithms reported in the literature. There may be only a few scenarios when complex more complex image analysis-based data association methods produce better quality for MOT at the cost of lowering speed (0.3 to 5 frames per second) [60]. To emulate the operation of such slow data association algorithms, the system performance for the proposed MOT-MR techniques is also compared by deliberately slowing down the processing speed of GDA to approximately 2 frames per second by using the *thread.sleep()* function in java. The thread running a map task is made to sleep for 500 milliseconds for every frame in the video sequence. Since only one thread is allocated on the CPU it remains

idle during the *sleep()* operation. The slowed GDA data association method is referred to as GDA_S.

4.2.4 Qualitative Performance

The qualitative performance of the video datasets that reflect the accuracy of tracking is analyzed by using the well-known performance metrics presented in [58]. These performance metrics have been discussed in the Section 3.8.2. The overall accuracy of tracking algorithms is aptly captured by MOTA (which combines FPs, FNs, and IDS). Hence the MOTA performance metric for MOT-MR is discussed next.

Figs. 4.3, 4.4 and 4.5 show the graphs comparing the MOTA achieved with the proposed MOT-MR techniques deployed on a Hadoop cluster in an Amazon EC2 cloud when the number of nodes (N) is increased for the PETS, MOT4 and PETS-E video. Each graph has three lines representing the three techniques: partially parallel (PP), fully parallel (FP), and fully parallel with frame overlap (FPO). Note that the video file is split into C chunks, and C is always equal to N, as explained in Section 4.1.

For all the three sequences, irrespective of the number of nodes, MOTA for PP remains constant (at the value achieved on a single node). The reason is that PP only parallelizes the time-independent object detection stage and the data association stage (time-dependent) is executed sequentially on a single reduce task where data from all the frames is available. Thus, the accuracy of the MOT results achieved with PP is not affected by the splitting of the video file. In Figs. 4.3 and 4.4, as the number of nodes is increased, the techniques FP and FPO display small variations in the MOTA value. The maximum difference between the MOTA value achieved on a single node and the MOTA values for a given MOT-MR technique achieved with a larger number of nodes is 0.5% for FP and 0.7% for FPO (see

Figs. 4.3 and 4.4). Considering the factor that the small video sequences PETS (795 frames) and MOT4 (1050 frames) are split into 10 chunks each (having 80-110 frame per chunk), 0.7% decrease in qualitative performance is very small. More importantly, when a large video file (PETS-E having 15000 frames) is split into multiple chunks, the loss in qualitative performance due to an increase in the number of nodes is negligible. As shown in Fig. 4.5, when N is increased from 5 to 20, MOTA remains almost constant for both FP and FPO at a value achieved also by PP. A 0.1 % decrease in MOTA is observed for FP when N is changed from 15 to 20. It shows the effectiveness of the MOT-MR techniques especially when the video file is large (which is often the case in many real-world systems).

As shown in Figs. 4.3, 4.4 and 4.5, the difference in MOTA for FP and FPO for a given N is small, but for any given N, FPO demonstrates a better qualitative performance than FP (except when N=4 for the MOT4 sequence). This shows the importance of splitting a

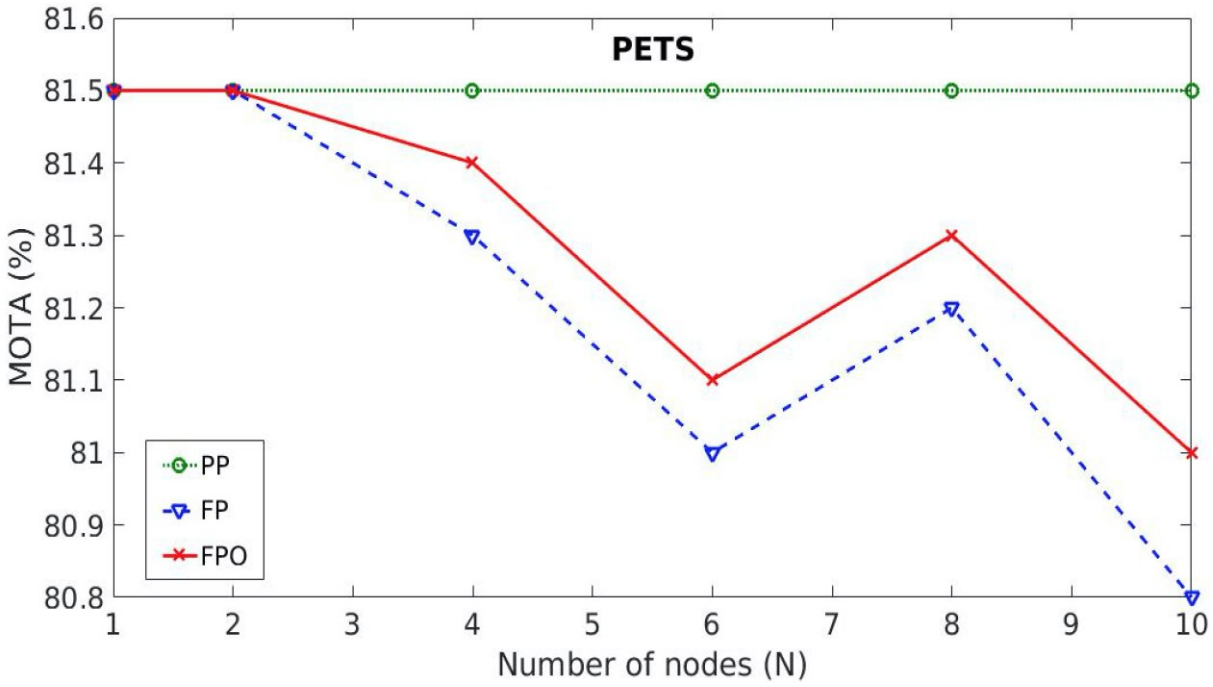

Figure 4.3: The effect of N on MOTA (%) for the PETS sequence.

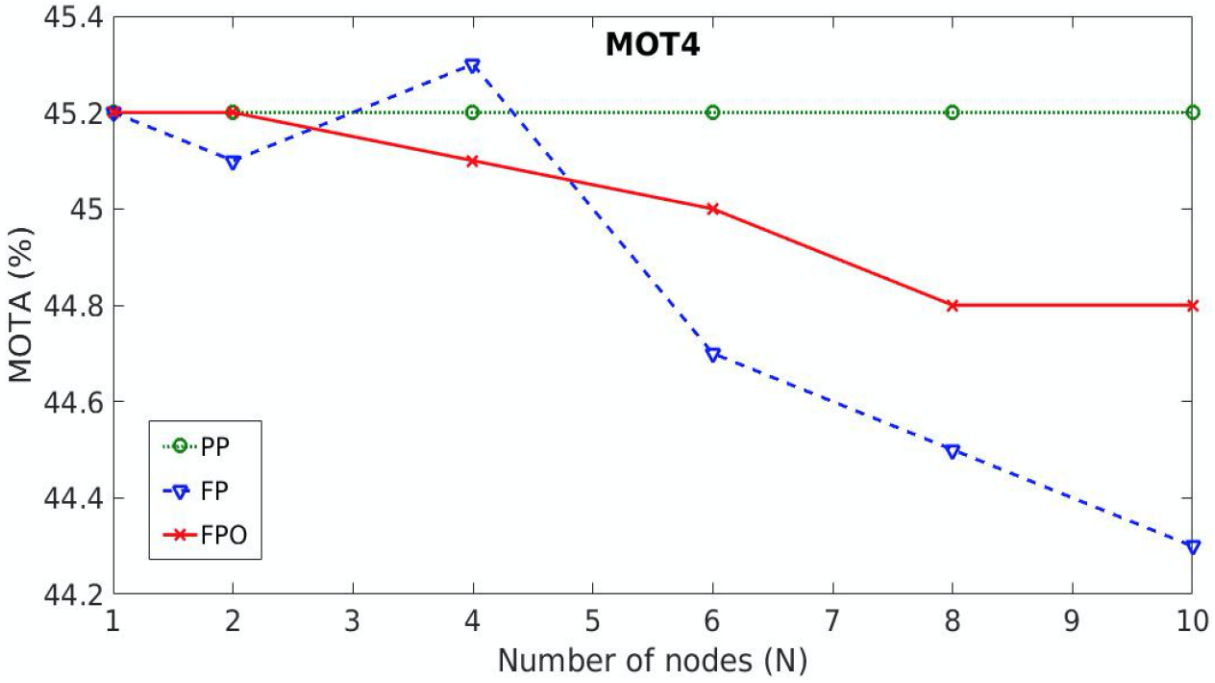

Figure 4.4: The effect of N on MOTA (%) for the MOT4 sequence.

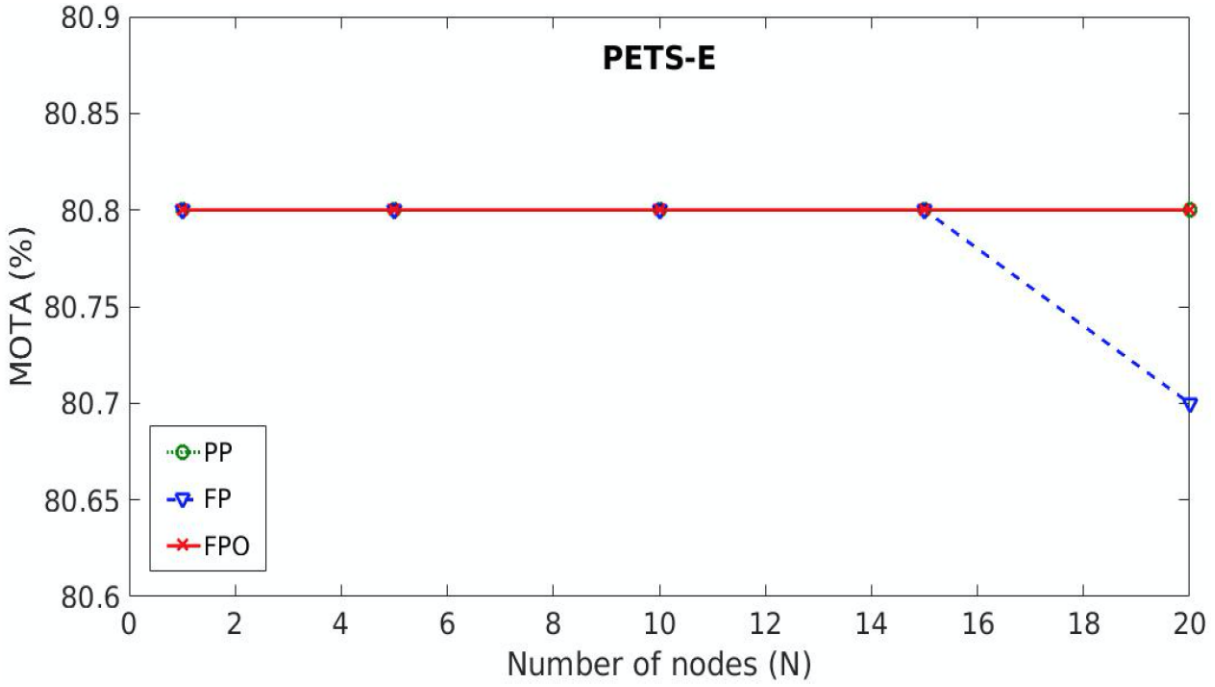

Figure 4.5: The effect of N on MOTA (%) for the PETS-E sequence. Lines PP, FP, FPO are overlapping

video file with overlapping frames. The overlapping frames provide motion and affinity measures of adjacent chunks, which helps improve the qualitative performance that reflects the accuracy of MOT.

4.2.5 System Performance

The system performance metrics that are used to compare the effectiveness and benefits of the three parallel MapReduce algorithms executing on a Hadoop cluster are presented next.

- **Computational Speedup (S_N)**: It is the ratio of the sequential execution time of the algorithm on a single node and the execution time on N nodes. $S_N=T_1/T_N$, where T_i is the execution time on i nodes [64].
- **System Efficiency (E_N)**: It is defined as average useful processor utilization with N processors and is given by $E_N=T_1/(N*T_N)$ where N is a number of processors [64].

4.2.5.1 Computational Speedup (S_N) with GDA

Figs. 4.6, 4.7 and 4.8 compare the S_N achieved on a multi-node Hadoop cluster deployed on Amazon EC2 cloud for video sequences PETS, MOT4 and PETS-E respectively when the GDA technique is used in the data association stage. For FP and PP, S_N increases steadily with N in all the three sequences. It shows the scalability of the proposed MOT-MR techniques. From the given N values, the maximum S_N achieved by PP is 7.74 when N=10, 9.03 when N=10 and 17.2 when N=20 for PETS, MOT4 and PETS-E respectively. The maximum S_N achieved by FP is 7.73 when N=10, 9.03 when N=10, and 17.4 when N=20 for PETS, MOT4, PETS-E respectively. Thus, the difference between S_N of FP and PP for a given video sequence is very small. The reason is that the data association method GDA is very fast and running it in parallel does not provide a noticeable improvement in S_N. Since

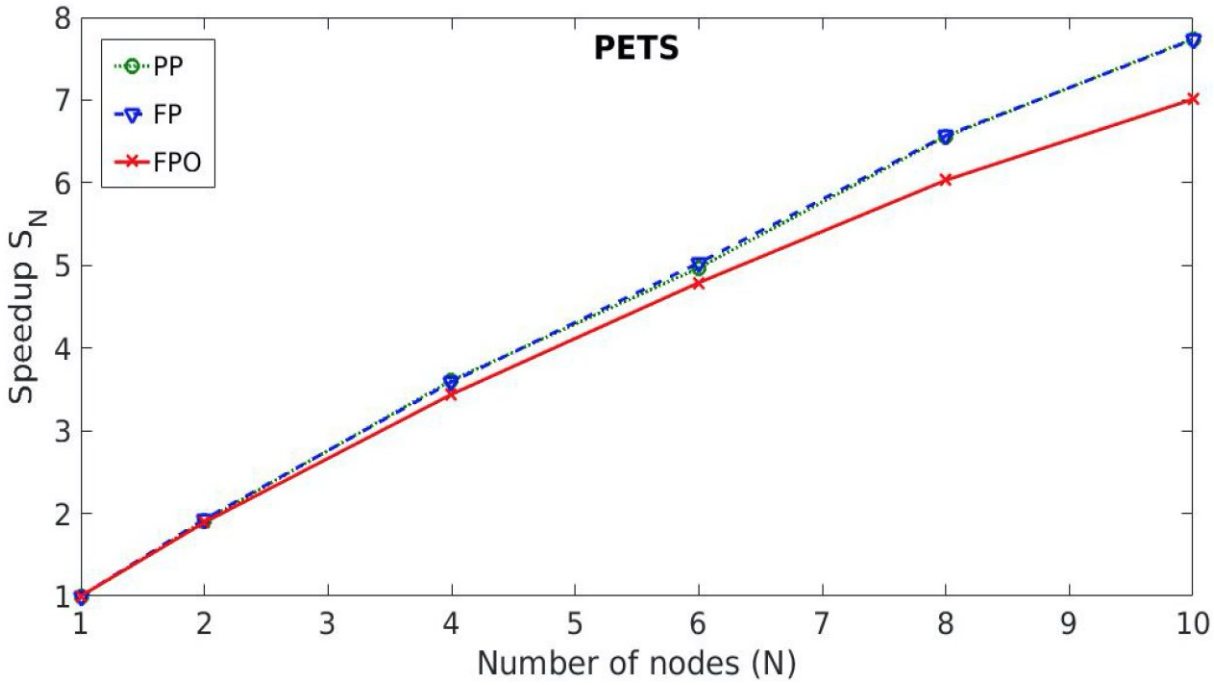

Figure 4.6: The effect of N on S_N using GDA for the PETS sequence.

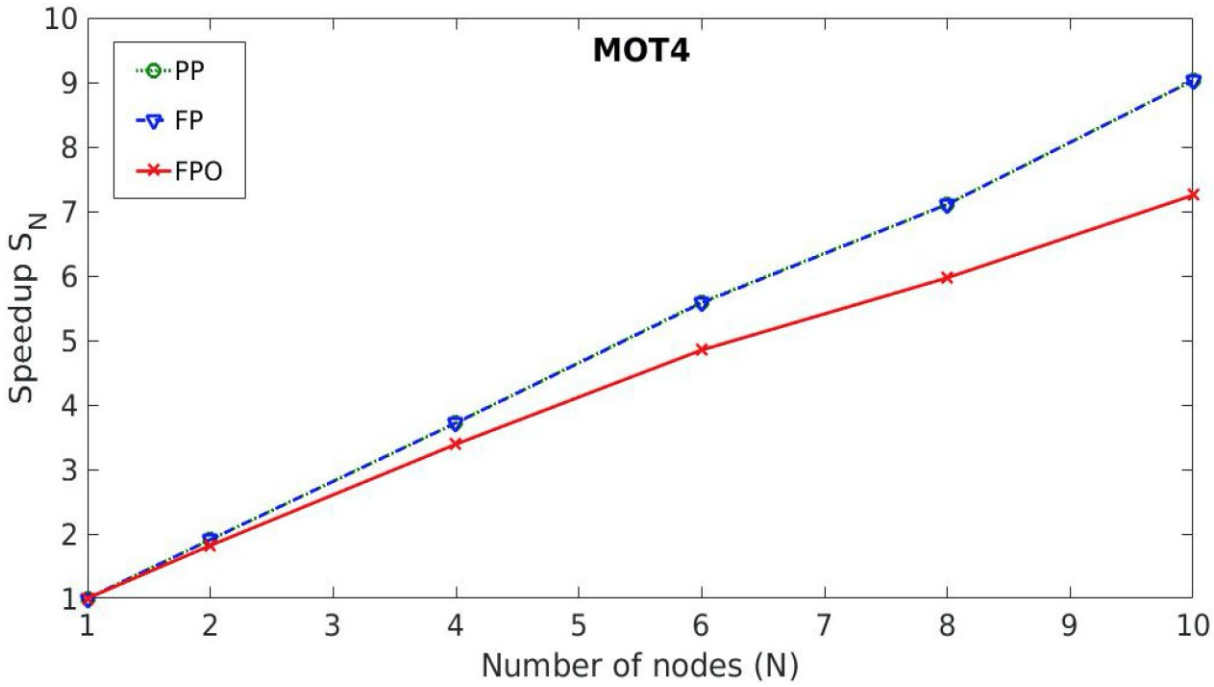

Figure 4.7: The effect of N on S_N using GDA for the MOT4 sequence.

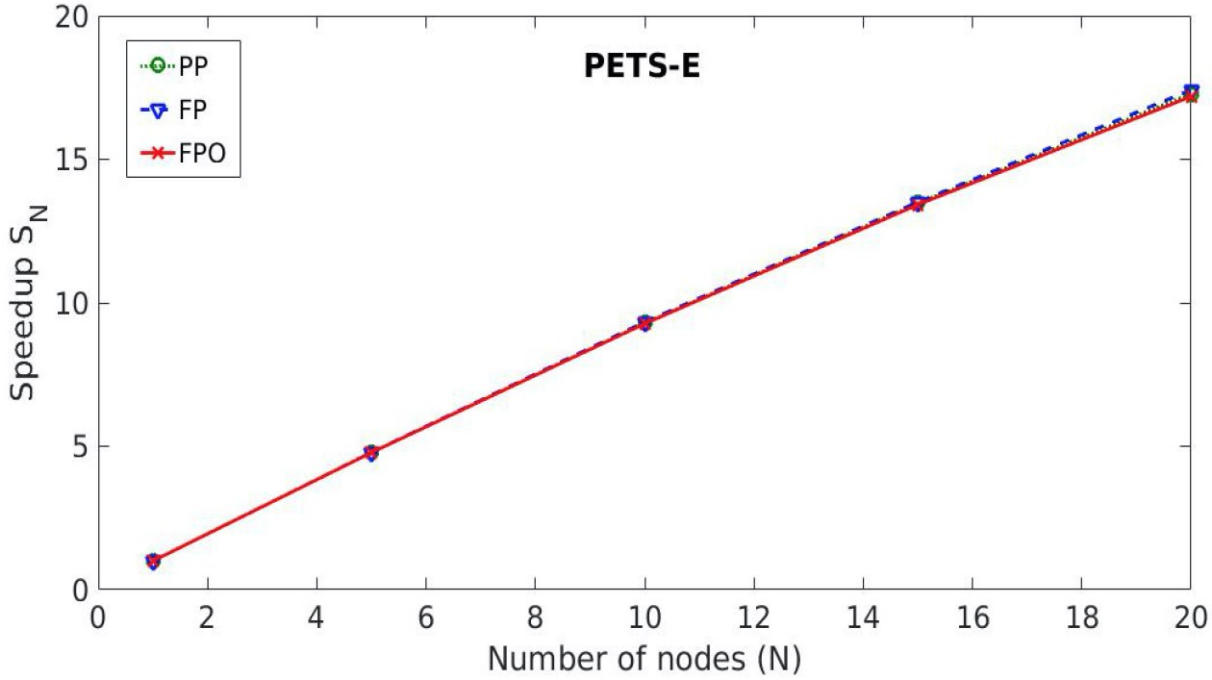

Figure 4.8: The effect of N on S_N using GDA for the PETS-E sequence.

PP leads to the best qualitative performance and S_N achieved by PP is almost the same as compared to FP, it becomes a preferable choice for MOT-MR when a fast data association technique is available. For any N, FPO produces a lower S_N in comparison to FP for all the three video sequences. The reason is that due to frame overlapping, FPO has an additional overhead of processing the extra δ frames.

4.2.5.2 Computational Speedup (S_N) with GDA_S

The comparison of S_N achieved by the proposed techniques while using GDA_S is given by Figs. 4.9, 4.10 and 4.11 for PETS, MOT4 and PETS-E video sequences respectively. It can be seen that FP has a higher S_N than PP in all the three sequences. It is because, in PP, the data association stage is executed sequentially in a single reduce task whereas, in FP, the data association stage is parallelized. When the slower data association method is used,

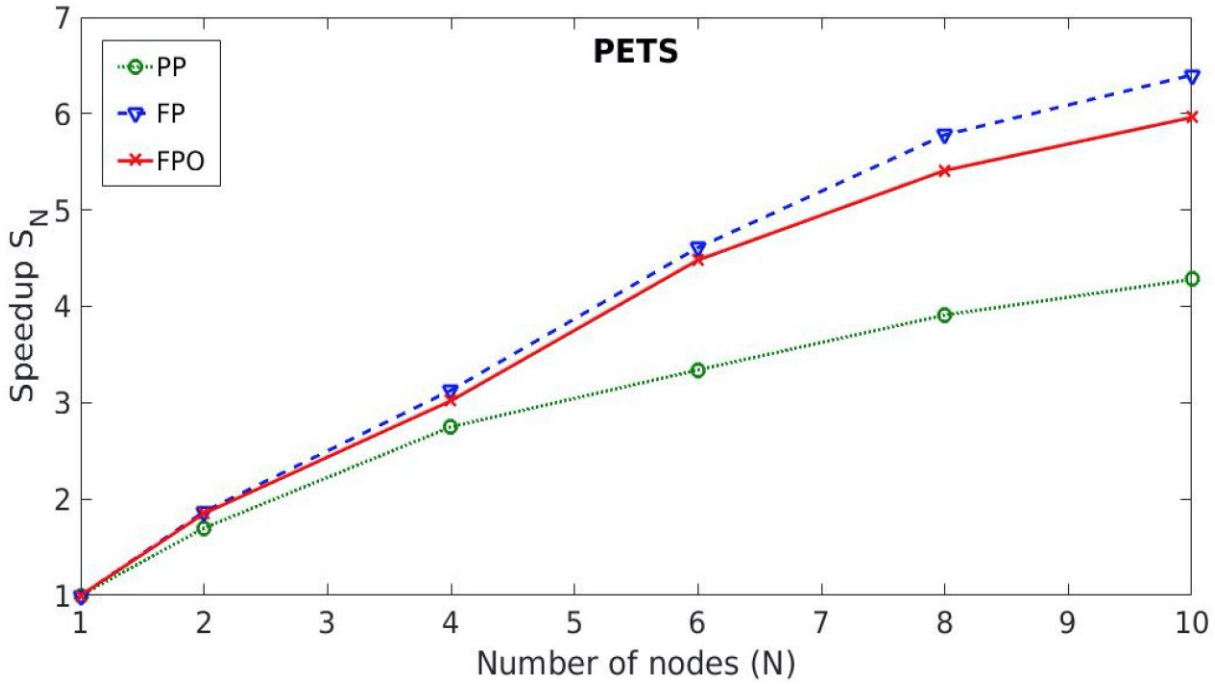

Figure 4.9: The effect of N on S_N using GDA_S for the PETS sequence.

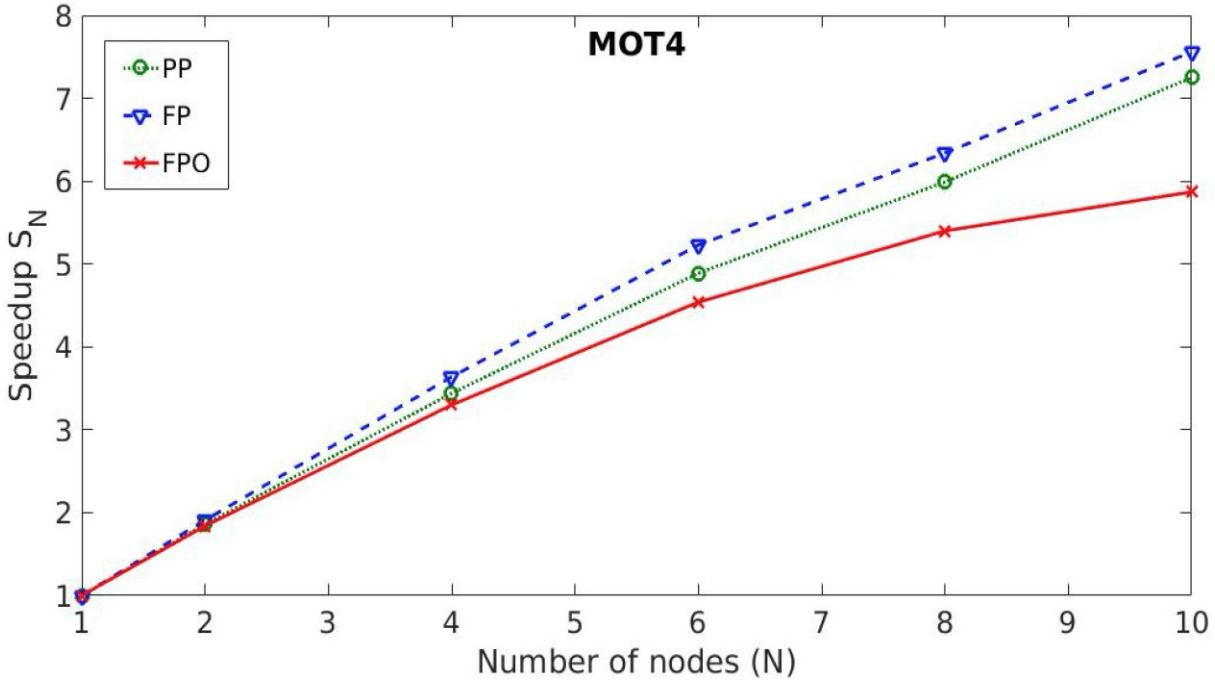

Figure 4.10: The effect of N on S_N using GDA_S for the MOT4 sequence.

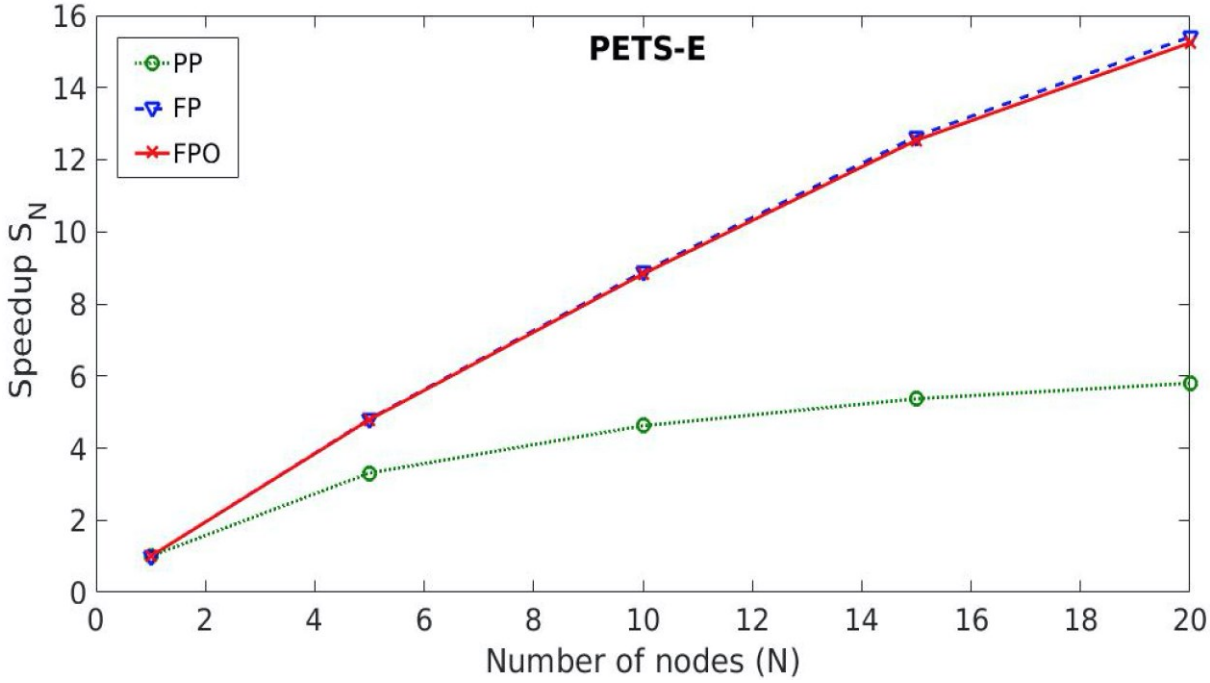

Figure 4.11: The effect of N on S_N using GDA_S for the PETS-E sequence.

the reduce task in PP becomes a bottleneck and S_N is reduced. The difference between S_N of PP and FP is large in the case of the PETS-E sequence because the video file is large. FP is able to achieve the S_N of 15.4 as compared to 5.8 of PP when N=20. This shows that FP is more scalable when slower data association method is used whereas, PP is not.

For a given N, FPO always has lower S_N than FP due to the frame overlapping overhead, but the difference in S_N between FPO and FP is small when the video file is large. FPO has a higher S_N than PP in the case of PETS and PETS-E sequences, whereas FPO has a lower S_N than PP for the MOT4 sequence. A higher value of δ=40 in MOT4 sequence increases the additional overhead of processing the overlapping frames, thus, S_N of FPO for the MOT4 sequences is lower than that of PP.

4.2.5.3 System Efficiency (S_N) with GDA

Figs. 4.12, 4.13, and 4.14 compares the efficiency, E_N, of the MOT-MR techniques for the PETS, MOT4 and PETS-E sequences respectively when GDA is used for data association. E_N decreases for all three techniques (PP, FP, FPO) as N is increased which is a typical scenario in many parallel systems. In all the three video sequences, the technique FPO produces the lowest efficiency for a given N. The FPO technique has frame overlapping which results in an additional overhead of processing a relatively higher number of frames in comparison to the total number of frames in the original video sequence. As a result, the useful utilization of the nodes decreases and thus E_N decreases. The lowest E_N for FPO is 70 %, and 72.5 % respectively for the PETS and MOT4 sequences achieved with N=10 (see Figs. 4.12, 4.13), whereas for the PETS-E sequence, the lowest E_N for FPO is 85.99 % achieved with N=20 (see Fig. 4.14). Among the three MOT-MR techniques, the highest efficiency is achieved by both FP and PP for the MOT4 sequence when N is larger than 1 (see Fig 4.13). The lowest E_N of 90% is achieved by these two techniques when N=10. Similarly, FP and PP achieves the highest efficiency of 86.4 % and 87.1 % respectively for the PETS-E sequence when N=20. Overall, for a given N, the techniques FP and PP display comparable speedups for each of the three video sequences. The reason is that the GDA technique is significantly faster and parallelizing it does not have much impact on the efficiency. Since the PP technique does not deteriorate the tracking accuracy, it is a preferable choice when the fast data association technique is used.

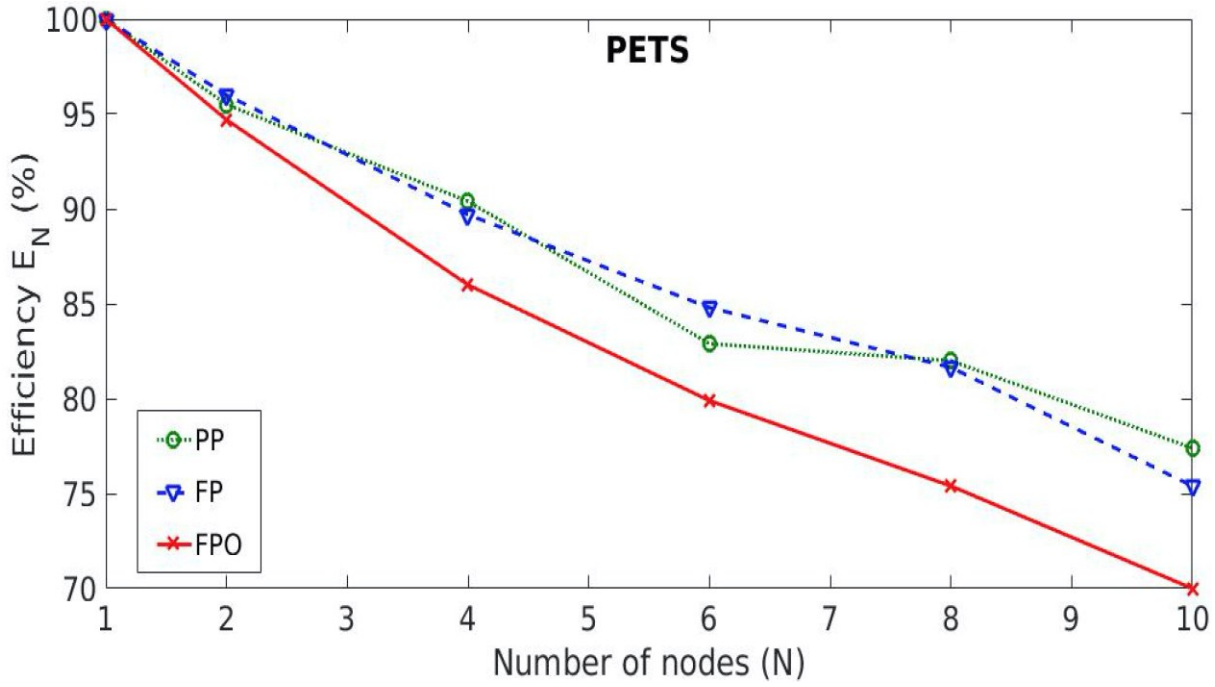

Figure 4.12: The effect of N on E_N for the PETS using GDA.

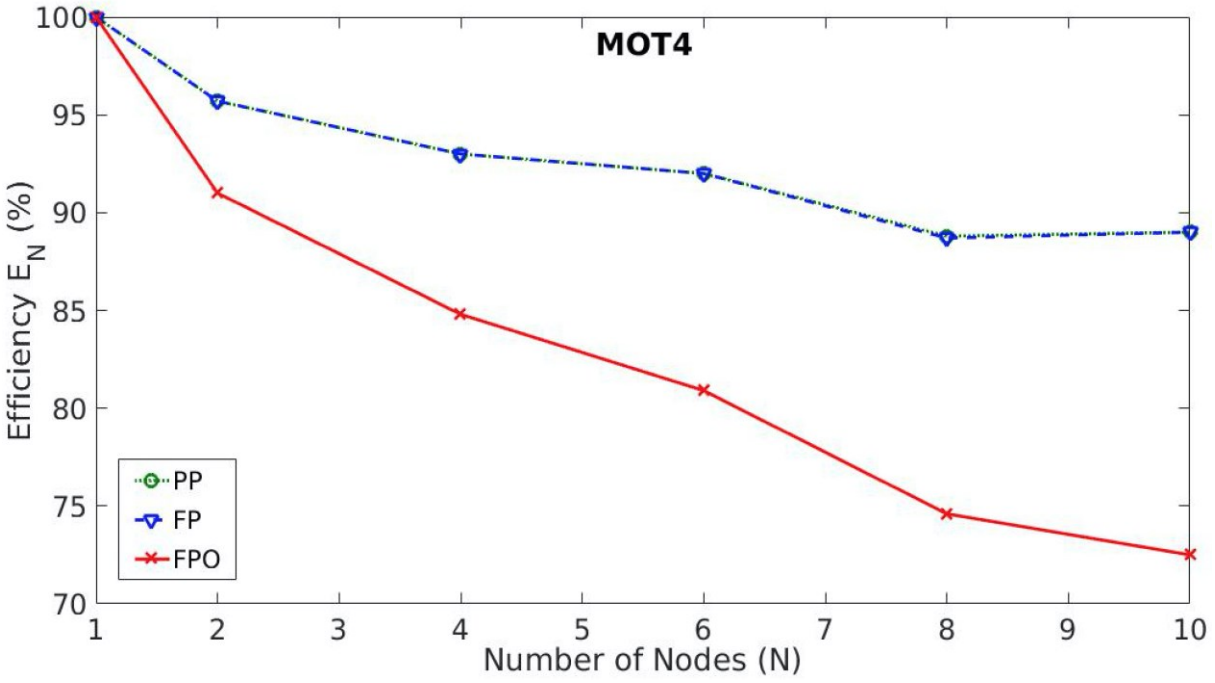

Figure 4.13: The effect of N on E_N for the MOT4 using GDA.

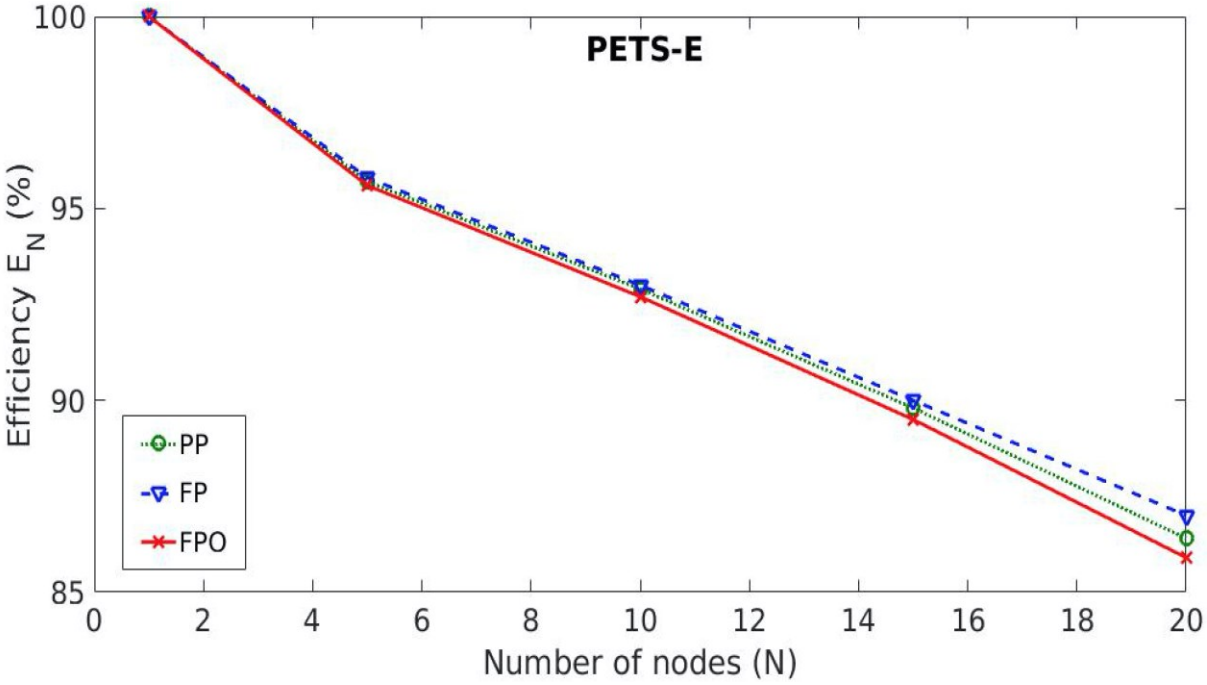

Figure 4.14: The effect of N on E_N for the PETS-E using GDA.

4.2.5.4 System Efficiency (E_N) with GDA_S

Figs. 4.15, 4.16, and 4.17 compares E_N for the three MOT-MR techniques achieved with the PETS, MOT4 and PETS-E sequences respectively when the slow data association, GDA_S, is used. Similar to GDA, for all three techniques (PP, FP, FPO), E_N decreases with increase in N. When the slow data association technique is used, for a given N, the technique FP achieves the highest efficiency, E_N, when compared with the other two techniques PP and FPO. FP has higher E_N as compared to PP because FP parallelizes both the object detection and the data association stages, whereas PP only parallelizes the object detection stage. Similar to the system in which a fast data association (GDA) is used, FP

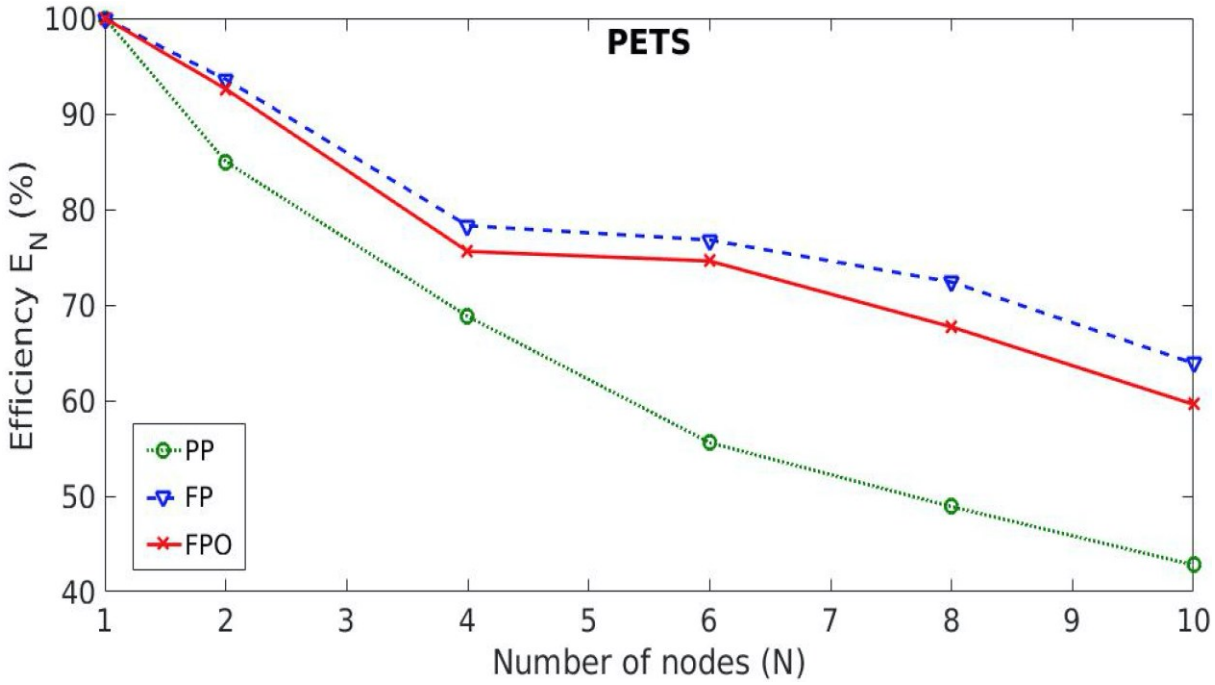

Figure 4.15: The effect of N on E_N for the PETS using GDA_S.

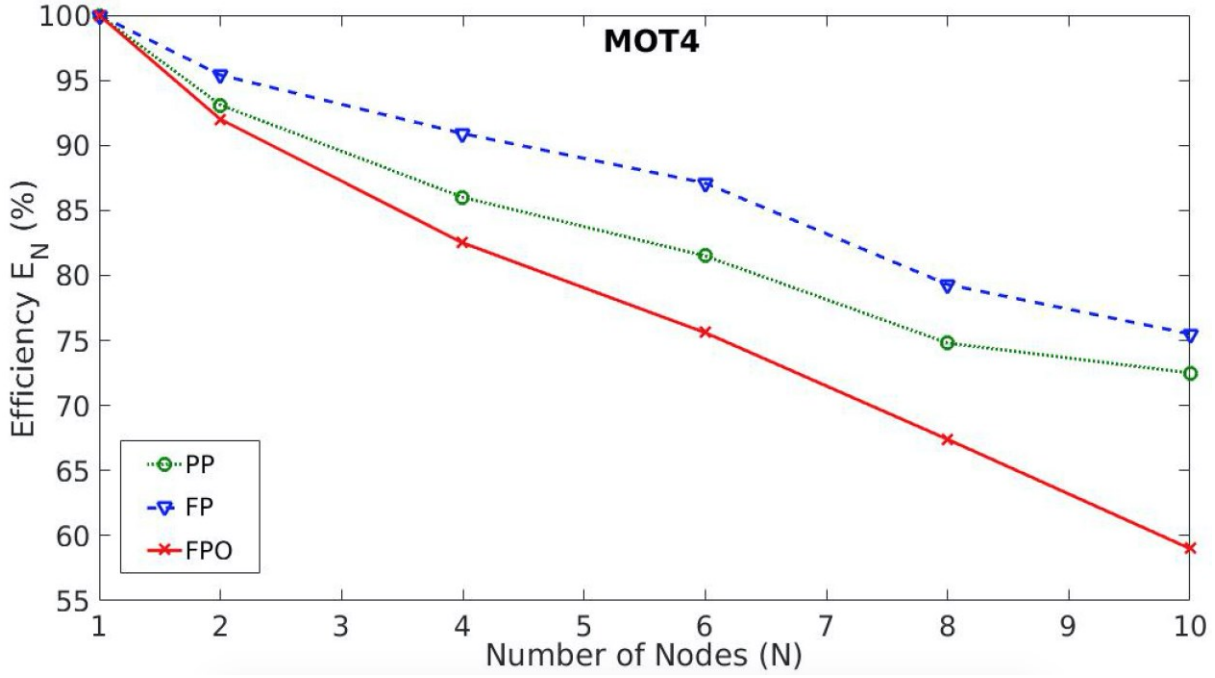

Figure 4.16: The effect of N on E_N for the MOT4 sequence using GDA_S.

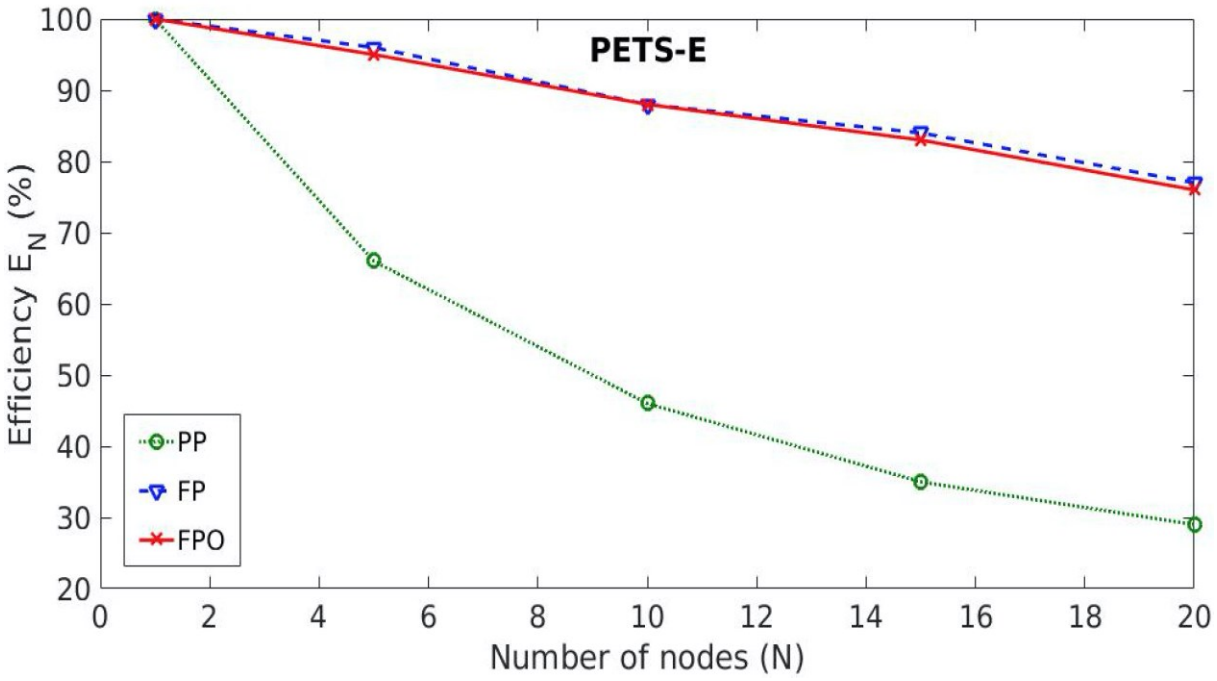

Figure 4.17: The effect of N on E_N for the PETS-E sequence using GDA_S.

leads to a higher E_N than FPO for any given N because FPO has an additional overhead as it processes an additional number of frames due to frame overlapping.

For the large video sequence, PETS-E, PP has a significantly lower E_N, only 29 % as compared to 77 % and 76.2 % achieved by FP and FPO respectively when N=20. It shows the limited scalability of PP the large video sequences. When the fast data association technique is used, technique FP is a preferable choice because it has the highest efficiency for both the small and large video sequences. Also, the tracking accuracy of FP and FPO are almost similar, especially when the video sequence is large (see Fig. 4.5).

FPO has a higher E_N for any given N as compared to PP for the PETS and PETS-E sequences whereas PP has higher E_N as compared to FPO in the case of the MOT4 sequence. For the MOT4 sequence, a larger value of δ (=40) has a higher overhead of processing an additional number of frames which lowers E_N of FPO.

4.2.6 Impact of δ on performance.

Experiments with all possible values of N (1 to 20), the three video sequences (PETS, MOT4, PETS-E), and the two GDA techniques (GDA, GDA_S) will give rise to a large number of experiments. In order to reduce the cost and time associated with experiments that are run on the AWS EC2 cloud, the investigation of the impact of δ on system performance is performed only for the PETS sequence for a single value of N=10. The results are presented in Table 4.1. The system performance deterioration for FPO in comparison to FP increases with an increase in δ. As shown in Table 4.1, varying δ=10 to 20 to 30 and using N=10 leads to approximately a 9.3%, 16.6%, and 23.3% increase in the difference between the S_N achieved by FP and FPO respectively. Similarly, with an increase in δ, the difference between E_N of FP and FPO increases.

Table 4.1: The effect of δ on S_N and E_N for FPO when N=10 and GDA_S is used

δ	S_N for FP	S_N for FPO	E_N (%) for FP	E_N (%) (for FPO)
10	7.74	7.01	77.4	70.1
20	7.74	6.46	77.4	64.6
30	7.74	5.93	77.4	59.3

4.3 Summary

This chapter introduces three different MapReduce-based parallel techniques: partially parallel (PP), fully parallel (FP), and fully parallel with frame overlap (FPO). The efficacy of the techniques is demonstrated through prototyping and measurement performed on the Amazon EC2 cloud. Insights resulting from the performance evaluation are summarized below.

- **Tracking Accuracy**: Irrespective of the value of N and the data association method used, the PP technique always leads to the best tracking accuracy.
- **Fast data association method**: PP is better because its qualitative performance (accuracy) does not deteriorate with the number of nodes. Moreover, for any given N, the S_N for PP is comparable to the S_N achieved by FP. Also, for any given N, the S_N of FPO is observed to be the lowest (see Figs. 4.3, 4.4, 4.5).
- **Slow data association method and small video files**: FP is better than PP and FPO in terms of S_N. PP has a slightly better qualitative performance (MOTA) than both FP and FPO. The improvement in MOTA achieved by FPO over FP is small and the difference in MOTA depends on N and δ. For a given N, FPO tends to have a lower S_N than FP.
- **Slow data association method and large video files**: FP is better than PP by a large margin in terms of S_N. The difference in qualitative performance between FP and PP is negligible for any given N. For any given N, FPO has a slightly better qualitative performance than FP whereas, for a given N, FP has a slightly better S_N than FPO. FP and FPO seem to be more scalable in comparison to PP. For example, when N=20, E_N of FP, FPO and PP is 77%, 76.2%, and 29% respectively (see Fig. 4.17).

Overall, the results of the experiments described in this chapter indicate that the fully parallel techniques outperform PP when slow data association methods are used in MOT. For a fast data association method, PP technique seems preferable.

Chapter: 5 Conclusions and Future Work

This chapter presents a summary of the thesis research, the conclusions and directions for future work directions.

5.1 Summary and Conclusions

In this thesis, fast and scalable techniques for Multiple Object Tracking (MOT) were presented. There are two major contributions of this thesis: a greedy data association technique (GDA) and the MOT-MR techniques. Each of these is discussed next.

5.1.1 GDA

The first contribution of this thesis is GDA, a greedy data association technique that iteratively associates the object detections by only using a linear motion model. There are two key steps in GDA. The first step, Tracklet Creation step, associates the object detections by using a Euclidean distance as a cost function, and creates a set of short object trajectories. The second step, Tracklet Association Step, associates the short object trajectories by utilizing the linear velocity of the objects to generate the final object trajectories. The performance analysis of the GDA technique performed on a desktop computer using a set of benchmark video sequences shows that GDA achieves a tracking accuracy (qualitative performance) comparable of the current state-of-the-art methods, and exhibits a computational speedup that is 50-600 times the speed achieved by most of the current state-of-the-art methods that produce a comparable qualitative performance. One of the drawbacks of GDA is that its tracking accuracy (qualitative performance) degrades considerably when a video sequence is captured with a fast-moving camera, for example, a camera installed on a bus. The moving camera introduces the non-linearity in the objects' motion and as GDA uses only linear velocity model, its performance degrades

considerably. With moving cameras, MOT can be handled better by using a non-linear motion model or/and a complex appearance model. The GDA technique is still able to perform well on a video sequence which is captured with a stationary camera and has a mixture of linear and non-linear object motions. As long as the nonlinearity in the video can be handled using piece-wise linear model, GDA will be able to provide a qualitative performance that is comparable to that produced the state-of-the-art techniques described in the literature.

5.1.2 The MOT-MR Techniques

The second contribution of this thesis is the MapReduce-based parallel techniques for MOT. The GDA technique improves the computational speed only for the data association stage of MOT. By parallelizing the end-to-end MOT that includes all its stages such as object detection and data association, the MOT-MR techniques enhance the overall speed of MOT. MOT has two stages, object detection and data association. The object detection stage has no time dependencies whereas the data association stage needs to handle time-dependencies because it utilizes the affinity measures from the adjacent frames. To handle the time-dependencies, three different MOT-MR techniques have been proposed: Partially Parallel, PP; Fully Parallel, FP; and Fully Parallel with Frame Overlap, FPO. The PP technique parallelizes the object detection stage during the Map phase of MapReduce and runs the data association stage sequentially during the Reduce phase. The FP parallelizes both the object detection and data association stages during the Map phase of the MapReduce and then combines the object trajectories, obtained from each map task, during the Reduce Phase. FPO is similar to FP except that it uses the frame overlapping while

splitting the video. The frame overlapping provides the affinity measures from the adjacent video chunks that enhance the qualitative performance of MOT.

The performance analysis of the proposed MOT-MR techniques was demonstrated by running the experiments on a Hadoop Cluster deployed on the AWS EC2 cloud. Three benchmark video sequences (PETS, MOT4, PETS-E) were used for the evaluation of the MOT-MR techniques. Two versions of the GDA technique were used: the fast data association (GDA) and the slow data association (GDA_S). The following conclusions are made from the results of the experiments performed.

- The PP technique is preferable when the fast data association method is used. The PP technique does not deteriorate the tracking accuracy (qualitative performance) because it does not parallelize the operations in the data association stage. Also, for a given N, the PP technique achieves similar speedup and efficiency as compare to the FP and FPO, when a fast data association technique is used.
- When the slow data association technique is used, the performance of PP deteriorates as the number of nodes is increased. The FP technique becomes preferable because it achieves the highest speedup for a given N, without noticeable deterioration in tracking accuracy, especially when the large video sequence is used.
- The FPO technique improves the tracking accuracy in comparison to FP, but has a slightly lower speedup and efficiency, especially when the small video sequences are used. When the large video sequence is used, the tracking accuracy and the system performance of FP and FPO are almost the same.

- Overall, the MOT-MR techniques provide scalable solutions for MOT. The maximum efficiency of 87% and 77% are achieved when the fast data association and slow data association method is used respectively for the large video sequence, PETS-E, when N=20. It shows that the proposed MOT-MR techniques are easily scalable.

5.2 Future Work

Directions for future research includes the following.

- Hybrid multiple object tracking methods form an important future work direction. For such a method, the data association results can be initially generated by faster techniques such as GDA, and finally, these results can be improved by incorporating some of the existing state-of-the-art appearance model-based methods. The hybrid methods can possibly improve the tracking results both qualitatively and computationally. Furthermore, hybrid methods may be able to handle nonlinearities in the motion better than the proposed GDA technique
- The proposed GDA technique requires a manual initialization of the various parameters, that include $\theta_1^A, \theta_2^A, \theta_{max}^A, \theta_1^C, \theta_2^C, \theta_{max}^C, G_{max}, \alpha_1, \alpha_2$, based on the training video sequences. A machine learning-based approach, using reinforcement learning, for example, can be investigated. The method will focus on determining the various parameters by automatically learning from the training video sequences.
- Multiple object tracking often needs to be performed in a short period of time. An automatic technique for estimating the required number of nodes of a Hadoop cluster for performing MOT for a given video before the expiry of a user specified deadline warrants investigation.

- Extension of the MOT-MR techniques for handling streaming data from live stream videos, on a stream processing platform such as Apache Spark forms an interesting direction for future research.

References

[1] A. Hampapur, L. Brown, J. Connell, A. Ekin, N. Haas, M. Lu, H. Merkl and S. Pankanti, "Smart Video Surveillance: Exploring the Concept of Multiscale Spatiotemporal Tracking," *IEEE Signal Processing Magazine,* vol. 22, no. 2, pp. 38-51, 2005.

[2] D. Beymer, P. McLauchlan, B. Coifman and J. Malik, "A Real-Time Computer Vision System for Measuring Traffic Parameters," *in Proceedings of IEEE Conference on Computer Vision and Pattern Recognition,* pp. 495-501, 1997.

[3] A. Petrovskaya and S. Thrun, "Model Based Vehicle Detection and Tracking for Autonomous Urban Driving," *Autonomous Robots,* vol. 26, no. 2-3, pp. 123-139, 2009.

[4] M. Vincze, M. Schlemmer, P. Gemeiner and M. Ayromlou, "Vision for Robotics: a Tool for Model-based Object Tracking," *IEEE Robotics & Automation Magazine,* vol. 12, no. 4, pp. 53-64, 2006.

[5] J. Lautissier, L. Legrand, A. Lalande, P. Walker and F. BrunotteI, "Object Tracking in Medical Imaging using a 2D Active Mesh System," *in Proceedings of IEEE Conference on Engineering in Medicine and Biology Society,* pp. 739-742, 2003.

[6] J. Berclaz, F. Fleuret, E. Turetken and P. Fua, "Multiple Object Tracking Using K-Shortest Paths Optimization," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 33, no. 9, pp. 1806-1819, 2011.

[7] P. F. Felzenszwalb, R. B. Girshick, D. McAllester and D. Ramanan, "Object Detection with Discriminatively Trained Part-Based Models," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 32, no. 9, pp. 1627-1645, 2010.

[8] P. Felzenszwalb, D. McAllester and D. Ramanan, "A Discriminatively Trained, Multiscale, Deformable Part Model," *in Proceedings of IEEE Conference on Computer Vision and Pattern Recognition,* pp. 1-8, 2008.

[9] R. Girshick, F. Iandola, T. Darrell and J. Malik, "Deformable Part Models are Convolutional Neural Networks," *in Proceedings of IEEE Conference on Computer Vision and Pattern Recognition,* pp. 437-446, 2015.

[10] C.-H. Kuo, C. Huang and R. Nevatia, "Multi-target Tracking by On-line Learned Discriminative Appearance Models," *in Proceedings of IEEE Conference on Computer Vision and Pattern Recognition,* pp. 685-692, 2010.

[11] S. Tang, B. Andres, M. Andriluka and B. Schiele, "Subgraph Decomposition for Multi-Target Tracking," *in Proceedings of IEEE Conference on Computer Vision and Pattern Recognition,* pp. 5033-5041, 2015.

[12] A. Milan, S. Roth and K. Schindler, "Continuous Energy Minimization for Multitarget Tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 36, no. 1, pp. 58-72, 2014.

[13] B. Yang and R. Nevatia, "Multi-Target Tracking by Online Learning of Non-linear Motion Patterns and Robust Appearance Models," *in Proceedings of IEEE Conference on Computer Vision and Pattern Recognition,* pp. 1918-1925, 2012.

[14] R. Girshick, J. Donahue, T. Darrell and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," *in Proceedings of IEEE Conference on Computer Vision and Pattern Recognition,* pp. 580-587, 2014.

[15] G. Singh, S. Rajan and S. Majumdar, "A Greedy Data Association Technique for Multiple Object Tracking," *in Proceedings of IEEE International Conference on Multimedia Big Data,* 2017.

[16] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," *Sixth Symposium on Operating System Design and Implementation,* pp. 137-149, 2004.

[17] C. Ryu, D. Lee, M. Jang and E. Seo, "Extensible Video Processing Framework in Apache Hadoop," *in Proceedings of IEEE International Conference on Cloud Computing Technology and Science,* pp. 305-310, 2013.

[18] M. Yamamoto and K. Kaneko, "Parallel Image Database Processing with MapReduce and Peformance Evaluation in Pseudo Distributed Mode," *International Journal of Electronic Commerce Studies,* vol. 3, no. 2, pp. 211-228, 2012.

[19] T. Abdullah, A. Anjum, M. F. Tariq, Y. Baltaci and N. Antonopoulos, "Traffic Monitoring Using Video Analytics in Clouds," *in Proceedings of IEEE Confernece on Utility and Cloud Computing,* pp. 39-48, 2014.

[20] D. B. Reid, "An Algorithm for Tracking Multiple Targets," *IEEE Transactions on Automatic Control,* vol. 24, no. 6, pp. 843-854, 1979.

[21] A. Bewley, L. Ott, F. Ramos and B. Upcroft, "ALExTRAC: Affinity Learning by Exploring Temporal Reinforcement within Association Chains," *in Proceedings of IEEE International Conference on Robotics and Automation,* pp. 2212-2218, 2016.

[22] B. Yang and R. Nevatia, "Online Learned Discriminative Part-Based Appearance Models for Multi-Human Tracking," *in Proceedings of European Conference on Computer Vision,* pp. 484-498, 2012.

[23] W. Choi, "Near-Online Multi-target Tracking with Aggregated Local Flow Descriptor," *in Proceedings of IEEE Conference on Computer Vision,* pp. 3029-3037, 2015.

[24] F. Solera, S. Calderara and R. Cucchiara, "Learning to Divide and Conquer for Online Multi-Target Tracking," *in Proceedings of IEEE Conference on Computer vision,* pp. 4373-4381, 2015.

[25] B. Yang and R. Nevatia, "An Online Learned CRF Model for Multi-target Tracking," *in Proceedings of IEEE Conference on Computer Vision and Pattern Recognition,* pp. 2034-2041, 2012.

[26] D. Comaniciu, V. Ramesh and P. Meer, "Kernel-based Object Tracking," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 25, no. 5, pp. 564-577, 2003.

[27] A. Adam, E. Rivlin and I. Shimshoni, "Robust Fragments-based Tracking using the Integral Histogram," *in Proceedings of IEEE Conference on Computer Vision and Pattern Recognition,* pp. 798-805, 2006.

[28] G. R. Bradski, "Real Time Face and Object Tracking as a Component of a Perceptual User Interface," *IEEE Workshop on Applications of Computer Vision,* pp. 214-219, 1998.

[29] W. Hu, X. Li and W. Luo, "Single and Multiple Object Tracking Using Log-Euclidean Riemannian Subspace and Block-Division Appearance Model," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2012.

[30] L. Zhang and L. v. d. Maaten, "Structure Preserving Object Tracking," *in Proceedings of IEEE Conference on Computer Vision and Pattern Recognition,* pp. 1838-1845, 2013.

[31] M. Yang, T. Yu and Y. Wu, "Game-Theoretic Multiple Target Tracking," *in Proceedings of IEEE Conference on Computer Vision and Pattern Recognition,* pp. 1-8, 2007.

[32] R. Girshick, "Discriminatively trained deformable part models," 2012. [Online]. Available: http://people.cs.uchicago.edu/~rbg/latent-release5/. [Accessed February 2017].

[33] S. Ghemawat, H. Gobioff and S.-T. Leung, "The Google File System," *19th ACM Symposium on Operating Systems Principles,* pp. 29-43, 2003.

[34] "Apache Hadoop," April 2017. [Online]. Available: hadoop.apache.org.

[35] J. Black, T. Ellis and P. Rosin, "Multi View Image Surveillance and Tracking," *Motion and Video Computing Workshop,* pp. 169-174, 2002.

[36] T. Zhao and R. Nevatia, "Tracking Multiple Humans in Complex Situations," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 26, no. 9, pp. 1208-1221, 2004.

[37] R. Girshick, "Fast R-CNN," *in Proceedings of IEEE Conference on Computer Vision,* pp. 1440-1448, 2015.

[38] R. Girshick, "From Rigid Templates to Grammars: Object Detection with Structured Models", Ph.D. dissertation, Department of Computer Science, The University of Chicago, Illinois, USA, 2012".

[39] X. Zeng, W. Ouyang, B. Yang, J. Yan and X. Wang, "Gated Bi-directional CNN for Object Detection," *in Proceedings of European Conference on Computer Vision*, 2016.

[40] W. Ouyang, X. Wang, X. Zeng, S. Qiu, P. Luo, Y. Tian, H. Li, S. Yang, Z. Wang, C.-C. Loy and X. Tang, "DeepID-Net: Deformable Deep Convolutional Neural Networks for Object Detection," *in Proceedings of IEEE Conference on Computer Vision and Pattern Recognition,* pp. 2403-2412, 2015.

[41] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," *in Proceedings of IEEE Conference on Computer Vision and Pattern Recognition,* pp. 779-788, 2016.

[42] M. Rastegari, V. O. Ordonez, J. Redmon and A. Farhadi, "XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks," *in Proceedings of European Conference on Computer Vision,* 2016.

[43] H. Jiang, S. Fels and J. J. Little, "A Linear Programming Approach for Multiple Object Tracking," *in Proceedings of IEEE Conference on Computer Vision and Pattern Recognition,* pp. 1-8, 2007.

[44] A. Geiger, M. Lauer, C. Wojek, C. Stiller and R. Urtasun, "3D Traffic Scene Understanding from Movable Platforms," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 36, no. 5, pp. 1012-1025, 2014.

[45] H. Pirsiavash, D. Ramanan and C. C. Fowlkes, "Globally-optimal Greedy Algorithms for Tracking a Variable Number of Objects," *in Proceedings of IEEE Conference on Computer Vision and Pattern Recognition,* pp. 1201-1208, 2011.

[46] L. Zhang, Y. Li and R. Nevatia, "Global Data Association for Multi-object Tracking Using Network Flows," *in Proceedings of IEEE Conference on Computer Vision and Pattern Recognition,* pp. 1-8, 2008.

[47] A. A. Butt and R. T. Collins, "Multi-target Tracking by Lagrangian Relaxation to Min-cost Network Flow," *in Proceedings of IEEE Conference on Computer Vision and Pattern Recognition,* pp. 1846-1853, 2013.

[48] C. Huang, B. Wu and R. Nevatia, "Robust Object Tracking by Hierarchical Association of Detection Responses," *in Proceedings of European Conference on Computer Vision,* pp. 788-801, 2008.

[49] "Hungarian algorithm," [Online]. Available: https://en.wikipedia.org/wiki/Hungarian_algorithm. [Accessed April 2017].

[50] C. Kim, F. Li, A. Ciptadi and J. M. Rehg, "Multiple Hypothesis Tracking Revisited," *in Proceedings of IEEE Conference on Computer Vision,* pp. 4696-4704, 2015.

[51] S. Ren, K. He, R. Girshichk and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. PP, no. 99, pp. 1-1, 2016.

[52] J. Li, K. Lin and J. Wang, "Design of the Mass Multimedia Files Storage Architecture Based on Hadoop," *in Proceedings of International Conference on Computer Science and Education,* pp. 801-804, 2013.

[53] A. Garcia, H. Kalva and B. Furht, "A Study of Transcoding on Cloud Environments for Video Content Delivery," *ACM Multimedia Workshop on Mobile Cloud Media Computing,* pp. 13-18, 2010.

[54] M. Kim, Y. Cui, H. Seungho and H. Lee, "Towards Efficient Design and Implementation of a Hadoop-based Distributed Video Transcoding System in Cloud Computing Environment," *International Journal of Multimedia & Ubiquitous Engineering,* vol. 8, no. 2, 2013.

[55] M. Vaidya and S. Deshpande, "Study of Hadoop-based Traffic Management System," *in Proceedings of International Confernece on Recent Trends in Information Technology and Computer Science,* pp. 38-42, 2012.

[56] "Instance Types," [Online]. Available: https://aws.amazon.com/ec2/instance-types/. [Accessed April 2017].

[57] H. Tan and L. Chen, "An Approach for Fast and Parallel Video Processing on Apache Hadoop Clusters," *in Proceedings of IEEE International Conference on Multimedia and Expo,* pp. 1-6, 2014.

[58] A. Milan, L. Leal-Taixe, I. Reid, S. Roth and K. Schindler, "MOT16: A Benchmark for Multi-Object Tracking," in *arXiv:1603.00831 [cs.CV]*, 2016.

[59] F. J. and A. Shahrokni, "PETS2009: Dataset and Challenge," *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance,* pp. 1-6, 2009.

[60] A. Milan, S. Roth and K. Schindler, "MOTCHALLENGE," March 2017. [Online]. Available: https://motchallenge.net/.

[61] "FFMPEG-Formats," [Online]. Available: https://www.ffmpeg.org/ffmpeg-formats.html. [Accessed March 2017].

[62] "Batch Frame," [Online]. Available: http://www.batchframe.com/quick-tips/tip.php?t=3. [Accessed March 2017].

[63] L. Leal-Taixé, A. Milan, I. Reid, S. Roth and K. Schindler, "MOTChallenge 2015: Towards a Benchmark for Multi-Target Tracking," in *arXiv:1504.01942 [cs.CV]*, 2015.

[64] D. L. Eager, J. Zahorjan and E. D. Lazowska, "Speedup Versus Efficiency in Parallel Systems," *IEEE Transactions on Computers,* vol. 38, no. 3, pp. 408-423, 1989.

[65] "Multiple object tracking with prior detections and graph formalisms," April 2017. [Online]. Available: http://sites.uclouvain.be/ispgroup/index.php/Research/MultiObjectTracking.

[66] "Visual multiple object tracking in a video sequence," April 2017. [Online]. Available: https://www.youtube.com/watch?v=iXsAXufaWc8.