# Granulated RCNN and Multi-Class Deep SORT for Multi-Object Detection and Tracking

Anima Pramanik, Sankar K. Pal, *Life Fellow, IEEE*, J. Maiti, *Member, IEEE*, and Pabitra Mitra, *Member, IEEE*

***Abstract*—In this article, two new models, namely granulated RCNN (G-RCNN) and multi-class deep SORT (MCD-SORT), for object detection and tracking, respectively from videos are developed. Object detection has two stages: object localization (region of interest RoI) and classification. G-RCNN is an improved version of the well-known Fast RCNN and Faster RCNN for extracting RoIs by incorporating the unique concept of granulation in a deep convolutional neural network. Granulation with spatio-temporal information enables more accurate extraction of RoIs (object regions) in unsupervised mode. Compared to Fast and Faster RCNNs, G-RCNN uses (i) granules (clusters) formed over the pooling feature map, instead of its all feature values, in defining RoIs, (ii) only the positive RoIs during training, instead of the whole RoI-map, (iii) videos directly as input, rather than static images, and (iv) only the objects in RoIs, instead of the entire feature map, for performing object classification. All these lead to the improvement in real-time detection speed and accuracy. MCD-SORT is an advanced form of the popular Deep SORT. In MCD-SORT, the searching for association of objects with trajectories is restricted only within the same categories. This increases the performance in multi-class tracking. These characteristic features have been demonstrated over 37 videos containing single-class, two-class, and multi-class objects. Superiority of the models over several state-of-the-art methodologies is also established extensively, both qualitatively and quantitatively.**

***Index Terms*—Deep CNN, Foreground region proposal, Granulation, Object detection and tracking, Video analysis.**

## I. Introduction

MULTI-OBJECT detection and tracking in videos is an important task, and is used in a large number of application areas [1], including event detection, automatic driving, and robot navigation. Object detection and tracking can be done using either image processing (IP) techniques or machine vision system (MVS). Real-time videos consist of multiple objects. These videos are generated on-line in Electronic Engineering field using sensory system (camera). IP techniques are able to provide good results for image segmentation in videos, but it is restricted to simple traffic flow [1]. On the other hand, occluded multi-class objects cannot be classified using the IP techniques. These issues are addressed by MVS due to the vast improvement of deep convolutional neural networks (CNNs), benchmark datasets, and GPU's enhanced computing power. MVS joins machine learning (ML) algorithms with artificial intelligence (AI) for detecting and tracking multiple objects from real-time videos. Most advanced ML algorithm is deep learning that can generates and uses AI-features for detecting and tracking multi-objects in real-time videos. Therefore, deep learning is considered in this study.

Deep learning-based object detectors utilize deep CNNs as their backbone to extract features from the input images/video and classify the object(s) using these features. These detectors are categorized as i) two-stage detectors [2] and ii) one-stage detectors [3]. In two-stage detectors, first, approximate object(s) regions are proposed. Thereafter, features are extracted from the proposed region and used for the classification and bounding-box regression for the object candidate. Widely used region proposal methods are based on grouping super-pixels (e.g., selective search [4], POLWHE [5], etc.,), sliding windows (e.g., object in windows [6]), or wavelet transform (e.g., Kurtosis Wavelet Energy [7]). In one-stage detectors, on the other hand, bounding-boxes are predicted over input images without using region proposal. Two-stage detectors have high object localization and classification accuracy, whereas one-stage detectors achieve high inference speed [8]. Because of the availability of GPUs, two-stage detectors are considered as state-of-the-art detectors, of late, compared to one-stage.

A significant improvement in object detection using two-stage detectors has been achieved with the advancement of region-based deep CNNs, namely region-based CNN (RCNN) [9], Fast RCNN [10], and Faster RCNN [2]. These RCNNs are characterized by the concept of 'region proposal' based on pooling feature information in an image frame. Their performances depend on the accuracy in extracting the regions proposals. In RCNN and Fast RCNN, region proposal is done using the selective search method [4] that slows down their operation. Faster RCNN [2] replaces the selective search with a region proposal network (RPN), which is a fully CNN. All these two-stage detectors are found to be effective for object classification in images due to their proper training with million images from various benchmark datasets, including ImageNet [11] and PASCAL Visual Object Classes (VOC) [12]. An improvement over Faster RCNN is done in [13] for ship classification using Synthetic Aperture

Manuscript received March 9, 2020; revised May 29, 2020 and November 1, 2020; accepted November 15, 2020. Date of publication January 5, 2021; date of current version January 21, 2022. The work was supported in part by the UAY Project, GOI under Project Code: IITKGP-022. Funding agencies are Ministry of Education, Ministry of Steel, Govt. of India, and Tata Steel Ltd. India. *(Corresponding author: J. Maiti.)*

Anima Pramanik and J. Maiti are with the Industrial & Systems Engineering, Indian Institute of Technology Kharagpur, Kharagpur 721302, India (e-mail: apramanik17@gmail.com; jhareswar.maiti@hotmail.com).

Sankar K. Pal is with the Soft Computing, ISI, Kolkata 700108, India (e-mail: sankar@isical.ac.in).

Pabitra Mitra is with the Computer Science and Engineering, IIT Kharagpur, Kharagpur 721302, India (e-mail: pabitra@cse.iitkgp.ernet.in).

This article has supplementary downloadable material available at https://doi.org/10.1109/TETCI.2020.3041019, provided by the authors.

Digital Object Identifier 10.1109/TETCI.2020.3041019

Radar (SAR) images. But it is restricted to the simple traffic flow. However, all these detectors lack in dealing with temporal information which is essential in estimating the localization of object(s) in a video sequence; thereby making those less effective for object localization and classification in video. Moreover, since all these two-stage detectors use the entire pooling feature map as regions of interest (RoIs) for classification, some undesirable regions (e.g., background) might affect the detection accuracy for their wrong classification.

To address these issues, one may consider developing a mechanism in the said RCNNs that can estimate roughly the foreground regions (containing all possible objects) within the pooling feature map of deep CNN. Both spatial and temporal information of the entire feature map are used to estimate these foreground regions. The probability of occurrence of object candidates in those regions is higher than that in any other parts. These foreground regions containing either single or multiple object modules, and are called "granules". Processing with these granules, therefore, not only would obviate the exclusion of desirable regions, but also reduce the operation (search) space for further processing. This process reduces the cost-complexity in video object detection. This constitutes the underlying principle of the proposed granulated RCNN (G-RCNN) where the concept of optimum granulation is used in the pooling feature map. Accordingly, the architecture (e.g., deep CNN) of Faster RCNN gets modified.

Granulation is a process of formation and representation of granules, evolved through information abstraction and derivation of knowledge from data. A granule is defined as a clump (cluster) of indiscernible objects drawn together, for example, by likelihood, similarity, nearness, or functionality. Recently, the efficacy of the concept of granular computing has been demonstrated for efficient video object localization and tracking in various uncertain situations under both supervised and unsupervised modes [14], [15]. Spatio-color neighborhood granules based on region growing method provide superior performance in unsupervised detection and tracking of occluded and overlapped objects [14]. In our proposed G-RCNN, we have adopted the irregular shaped neighborhood granules in the pooling feature map using both spatial and temporal similarity to localize the object(s) region in video frames. The commonality between the spatial and temporal maps is considered as the optimum map (called RoI map), representing the foreground (i.e., object) modules in the video sequence. Then, class specific SVMs are used to classify these object modules.

After classification, these classified objects are used for tracking, which is a two-fold task. First, a representation model of possible trajectories corresponding to each object, detected in a video, is computed. Later, that model is used in global optimization problem to estimate the affinity among the detected objects across different frames in the same video. Available tracking models are categorized as: (i) appearance models [16], (ii) motion models [17], [18], and (iii) composite models [19], [20]. Appearance models focus on computing easy-to-track object features that encode appearances of local regions of objects or their bounding-boxes [16]. However, such features are not robust with respect to illumination variation and occlusions. Moreover, this model cannot discriminate the objects with high similarity. The motion models encode object dynamics to predict their locations in the future frames. Such motion models include submodular optimization (SOP) [17] and simple online real-time tracking (SORT) [18]. In SOP [17], first, low level track-lets are generated based on the overlapping criteria and minimum cost flow, and then, the correct track-let corresponding to the detected object is selected. In SORT [18], data association is done based on motion-based association metric. These two motion models fail in case of long inter-frame object occlusions and complex motions. Composite models for tracking, on the other hand, aim at striking a balance between motion and appearance modeling. However, it is hard to achieve such balance in real-world [19]. In a composite model, a proper data association is done for each target object based on both motion and appearance information. Such composite models include deep affinity network (DAN) [19] and deep SORT [20]. In DAN [19], two deep CNNs function jointly to learn target object's appearances and their affinities in a pair of video frames in an end-to-end fashion. In deep SORT [20], data association is done based on both motion and appearance-based association metrics. Another deep model, Siamese network [21] is widely used for multi-object tracking. All the aforesaid models are good, both in time and accuracy, for tracking one-class objects (domain specific), but not for multi-class objects (general scenario). To address this issue, we have proposed a multi-class deep SORT (called, MCD-SORT), where frame-by-frame association for both motion and appearance features is done only for the same-class target objects. These same-class objects are detected from the aforesaid granulated foreground modules (RoI map) as obtained from G-RCNN. Consideration of the same-class objects, instead of all-class objects, for frame-by-frame association, reduces the searching space and computation time, and enhances the tracking accuracy.

The effectiveness of our G-RCNN and MCD-SORT algorithms along with various comparisons has been demonstrated extensively over 37 videos containing eight different objects. These videos are taken from Multi-Object Tracking 15 (MOT15) challenge dataset [22], Urban Tracker [23], and a steel plant video data Video Analytics Road 19 (VAR19). VAR19 is a new real-life traffic data, not available elsewhere, unlike PASCAL Visual Object Classes (VOC) [12] and MOT15, which are benchmark data available on-line. The contributions and novelty of the investigation are summarized below.

i) A new deep CNN model, named, G-RCNN for multi-object detection has been introduced. This is an improved version of the widely used Fast RCNN [10] and Faster RCNN [2] providing superior performance. G-RCNN takes video directly as input and considers both spatial and temporal information, unlike Fast RCNN and Faster RCNN. Incorporating granulated layers using spatio-temporal information within the deep CNN (e.g., AlexNet [3]) architecture ensures better object(s) localization (RoIs). Classification task being dealt only with the objects present in RoIs, increases the detection accuracy significantly. Speed-wise, G-RCNN is superior to Fast RCNN, while comparable to Faster RCNN.

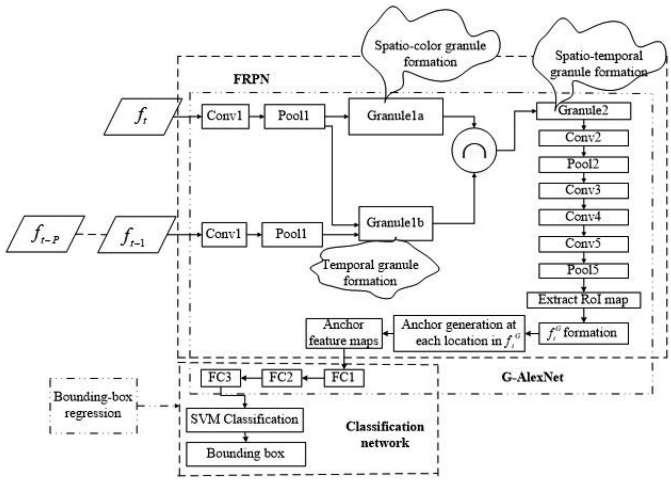


Fig. 1. G-RCNN Architecture

ii) MCD-SORT is a more efficient version of deep SORT [20] in terms of both accuracy and speed for tracking multiple objects, as it involves frame-by-frame association only within the same-class objects, instead of taking all classes together as one class.

iii) Besides the investigation demonstrates a way of incorporating the recently emerged granular computing in deep learning network for on-line object tracking and classification directly from input video. This is unique.

iv) VAR19 is prepared by us based on an Indian steel plant data. With class label information on a real-world scenario, VAR19 is significant, particularly, when the availability of real-life video data is scarce.

The article proceeds as follows: Section II presents the characteristics and training of the proposed G-RCNN. MCD-SORT is described in Section III. Results, including comparative studies are reported in Section IV. Finally, conclusion of the present study has been discussed in Section V.

## II. Proposed G-RCNN Method for Object Detection

Characteristics and training of the proposed G-RCNN are described in Sections II-A and II-B, respectively.

### A. G-RCNN

G-RCNN is an advanced version of Faster RCNN. A short description of RCNN and its variants is stated in Section B: S.I (see the supplementary material S.I). G-RCNN has two parts. The first part is called foreground region proposal network (FRPN), which is a granulated deep CNN that provides foreground RoIs over the video frame. Operation procedure of FRPN is briefly explained in Section II-A1. The second part is called classification network that deals with the classification of objects in each RoI. A brief description of classification network is stated in Section II-A2. Here, FRPN provides information to the second phase regarding the location where to explore for classification. The entire system is a single unified network, called G-RCNN as shown in Fig. 1. The feature generation network of G-RCNN, named G-AlexNet, is illustrated in the same figure also. It is the modification of AlexNet [3] architecture. AlexNet consists of five convolution (Conv1, Conv2, Conv3, Conv4, and Conv5), three pooling (Pool1, Pool2, and Pool5), and three fully connected (FC1, FC2, and FC3) layers. A short description of AlexNet architecture is stated in Section A: S.I (see the supplementary material S.I). As seen from Fig. 1, G-AlexNet consists of five convolution, three pooling, three granulation (Granule1$a$, Granule1$b$, and Granule2), one RoI-generation (RoI-map and $f_t^G$ formation), one anchoring process (anchor generation at each location in $f_t^G$ and anchor feature maps), and three fully connected layers. Granulation layers are incorporated after the Pool1 layer, and RoI-generation and anchoring process are done after the Pool5 layer of AlexNet to obtain the G-AlexNet. The activation function used is rectified linear unit [24].

*1) FRPN:* FRPN accepts the current video frame ($f_t$) having three channels (R, G, and B) and its previous $P$ frames ($f_{t-1}$ to $f_{t-P}$) as inputs. FRPN network passes these input frames through first convolution (Conv1) and pooling (Pool1) layers to extract their reduced feature maps. Thereafter, a set of spatio-color neighborhood granules (refer to Step 1) is computed on each frame at Pool1 feature map in Granule1$a$ layer. The network also computes temporal granules over the frames sequence of $f_t$, $f_{t-1}, \ldots, f_{t-P}$ at Pool1 map in Granule1$b$ layer. Then, these two kinds of granules are combined by keeping their commonality, resulting in spatio-temporal granules (refer to Step 2). This is done over Pool1 feature map of $f_t$ in Granule2 layer. After that, this spatio-temporal granuled feature map is passed through several convolution (e.g., Conv2, Conv3, Conv4, and Conv5) and pooling (e.g., Pool2 and Pool5) layers to generate its reduced map containing the RoIs (refer to Step 3). RoIs characterize the estimated objects region(s) in $f_t$. Therefore, anchoring process (refer to Step 4) is then performed at each RoI-pixel location for obtaining its approximate region proposal. In this process, several spatial windows of different scales and aspect ratios are drawn over each RoI-pixel location. These spatial windows are called anchors. Details of the operations are described below in step-wise.

*Step 1:* Formation of spatio-color neighborhood granules $\{G_{SCNG}\}$ in Granule1$a$ layer.

Granule1$a$ layer takes Pool1 feature map $\{f_t^{Pool1}\}$ as input and generates spatio-color neighborhood granules over it. Let $f_t$ be the current frame on which $\vartheta$ number of filters having size of ($W_{Conv1} \times W_{Conv1}$) are applied in the Conv1 layer to generate the feature map of $\vartheta$ number of channels. Thereafter, a filter size of ($W_{Pool1} \times W_{Pool1}$) is used on Pool1 layer to get the reduced feature map, say $\{f_t^{Pool1}\}$. Max pooling operation is done in this study to obtain this pooling feature map. Then, spatio-color quadrant granules $\{G_{SCG}\}$ of different sizes are obtained over all the channels of $\{f_t^{Pool1}\}$ using quad-tree decomposition [15] by considering both spatial and pooling feature similarity of pixels. All the spatio-color quadrant granules $\{G_{SCG}\}$, thus obtained, are mapped to the original $f_t$, resulting in $f_t^{G_{SCG}}$. In $f_t^{G_{SCG}}$, a $3 \times 3$ (minimum sized) window is moved and checked, if the pixels therein are of same intensity. Neighborhood (8-connected) windows of equal intensities are grouped by region growing technique; thereby forming different

spatio-color neighborhood quadrant granules $f_t^{G_{SCNG}}$ in the image frame $f_t$. Any two windows (grids), say, $g$ and $h$, in $f_t^{G_{SCNG}}$ are related by Eq. (1), as follows:

$$f_t^{G_{SCNG}} = \left\{ \begin{array}{l} (g,h) \in f_t^{G_{SCG}}, |(I(g) - I(h))| = 0 \ \& \\ (g,h) \text{ are } 8-\text{connected} \end{array} \right\} \quad (1)$$

where $I(.)$ represents the intensity of any pixel in grid (.). A spatio-color neighborhood quadrant granule is basically a collection of connected grids of same intensity. Therefore, its shape is arbitrary. The $f_t^{G_{SCNG}}$, so formed over $f_t$, can have multiple such granules of arbitrary shapes of different intensities, as well as granules of same intensity, but disconnected. The Pool1 features of $f_t^{G_{SCNG}}$ are called spatio-color neighborhood granules (say, $\{G_{SCNG}\}$) over $f_t^{Pool1}$ map in Granule1$a$ layer. $\{G_{SCNG}\}$ is then fed to the Granule2 layer to obtain the spatio-temporal granules.

*Step 2:* Formation of temporal $\{G_{TG}\}$ and spatio-temporal $\{G_{ST}\}$ granules in Granule1$b$ and Granule2 layers.

Temporal information of the input video sequence is used to obtain temporal granules $\{G_{TG}\}$ in the Granule1$b$ layer of G-AlexNet. Let $P$ be the number of previous frames of $f_t$ that are used as inputs to the FRPN. The FRPN computes a sequence of Pool1 feature maps $\{f_t^{Pool1}\}$, $\{f_{t-1}^{Pool1}\}$, ..., $\{f_{t-P}^{Pool1}\}$ corresponding to the inputs $f_t$, $f_{t-1}$, . . ., $f_{t-P}$. Note that each such Pool1 feature map consists of $\vartheta$ number of channels as determined by the number of filters/kernels used in the Conv1 layer of G-AlexNet. These feature maps corresponding to each channel are fed to the Granule1$b$ layer to generate temporal granules over the same channel using three point estimation [15] based on previous $P$ frames. These three points corresponding to each channel (say, $\varphi^{th}$ channel) are $P1 = max(f_{\varphi(t)}^{Pool1}, f_{\varphi(t-1)}^{Pool1}, ..., f_{\varphi(t-P)}^{Pool1})$, $P2 = median(f_{\varphi(t)}^{Pool1}, f_{\varphi(t-1)}^{Pool1}, ..., f_{\varphi(t-P)}^{Pool1})$, and $P3 = \min(f_{\varphi(t)}^{Pool1}, f_{\varphi(t-1)}^{Pool1}, ..., f_{\varphi(t-P)}^{Pool1})$. For each $\varphi^{th}$ channel, these points denote optimistic, most likely, and pessimistic features, respectively, and are used to estimate the background (mean) ($f_{\varphi(t)}^{Pool1}(\mu)$) and standard deviation ($\sigma_{\varphi(t)}^{Pool1}$) for the same channel of $\{f_t^{Pool1}\}$ [15]. These are defined as: $f_{\varphi(t)}^{Pool1}(\mu) = (P1 + 4 \times P2 + P3)/6$ and $\sigma_{\varphi(t)}^{Pool1} = (P1 - P3)/6$. Here, 'mean' represents the estimated background value for the current frame [15]. A location $(x, y)$ in $f_{\varphi(t)}^{Pool1}$ with value $f_{\varphi(t)}^{Pool1}(x, y)$ is detected as foreground if its difference from the mean ($f_{\varphi(t)}^{Pool1}(\mu)$) value is greater than $\sigma_{\varphi(t)}^{Pool1}$. Otherwise, the location $(x, y)$ is treated as a background. All these foreground features constitute what is called temporal granule(s) $G_{TG}^{\varphi}$ over that ($\varphi^{th}$) channel of $\{f_t^{Pool1}\}$. This is expressed as:

$$G_{TG}^{\varphi} = \left\{ \begin{array}{l} (x,y) \epsilon f_{\varphi(t)}^{Pool1}|, \\ |f_{\varphi(t)}^{Pool1}(x,y) - f_{\varphi(t)}^{Pool1}(\mu)| > \sigma_{\varphi(t)}^{Pool1} \end{array} \right\} \quad (2)$$

A set of temporal granules $\{G_{TG}\}$ formed over $\{f_t^{Pool1}\}$ is defined as:

$$\{G_{TG}\} = \left\{ G_{TG}^1 \cup G_{TG}^2 \cup ... \cup G_{TG}^{\varphi} \cup ... \cup G_{TG}^{\vartheta} \right\} \quad (3)$$

Both spatio-color neighborhood $\{G_{SCNG}\}$ and temporal $\{G_{TG}\}$ granules are combined with Eq. (4) to extract the spatio-temporal granules $\{G_{ST}\}$ over all the channels of $\{f_t^{Pool1}\}$ in Granule2 layer.

$$\{G_{ST}\} = \{G_{SCNG}\} \cap \{G_{TG}\} \quad (4)$$

Basically, $\{G_{ST}\}$ comprises the features of those pixels which are present in both $\{G_{SCNG}\}$ and $\{G_{TG}\}$. $\{G_{ST}\}$ represents the set of pooling features which characterize approximately the object model.

*Step 3:* RoI extraction from $\{G_{ST}\}$ in Pool5 feature map.

In this process, first, $\{G_{ST}\}$ map passes through several hidden layers (convolution and pooling layers) in G-AlexNet to obtain its reduced feature map (called $\{f_t^{Pool5}\}$). Thereafter, we select only those pixels whose corresponding features are present in both maps $\{f_t^{Pool5}\}$ and $\{G_{ST}\}$. These pixels together constitute the RoIs, i.e., the foreground (objects) regions, corresponding to $f_t$.

*Step 4:* Anchoring process.

Anchoring process is performed on each pixel of the RoI feature map (containing RoIs). The objective of this process is to determine its possibility to be in a class. Pixel location in RoI map, representing the object localization, are first projected on the original $f_t$ plane, resulting in $f_t^G$, say. At each such RoI-pixel location in $f_t^G$, we place masks of different scales and aspect ratios, called anchors. These anchors represent the foreground region proposal.

Let $l$ be the number of anchors used at each RoI-pixel location in $f_t^G$. In our experiment, we have used anchors of three different scales and three different aspect ratios, i.e., $l = 9$. These anchors are of regular shaped. Let $\psi_i \times \varrho_i$ denotes the size of $i^{th}$ anchor, which is constructed around a RoI-pixel location $(x, y)$ in $f_t^G$. Then, the coordinate of the left corner point of $i^{th}$ anchor is $(x - \psi_i/2,\ y - \varrho_i/2)$. The two-dimensional pixel-array, thus generated inside an anchor, is called the anchor feature map. The sequence of these anchor feature maps for all RoI-pixels is sent to the classification network (refer to Fig. 1) for determining the appropriate bounding-box(s), class label(s), and classification/recognition score(s) of detected object(s) in a frame, as explained in Section II-A2. The pseudo-code for RoI generation is defined in Algorithm 1: S.II (see the supplementary material S.II).

*2) Classification Network:* Classification network takes a sequence of anchor feature maps as input and generates classification score and appropriate bounding-box for the detected object. The classification task includes three steps, such as 'anchor classification', 'fitting bounding-box', and 'object recognition, as explained below in step-wise.

*Step 1:* Anchor classification.

In order to perform classification, each anchor map needs to be resized to that of the training images used in G-AlexNet. The resized 2-dimensional anchor map is then converted to 1-dimensional weighted array through FC1 and FC2 layers of G-AlexNet. Thereafter, this 1-dimensional array is passed through FC3 layer for SVM-classification and bounding-box generation. For each input anchor map, the SVM classification provides its most probable class with classification probability (say, anchor

score) as output. As mentioned before, each RoI-pixel has 9 anchors; thereby resulting 9 anchor scores corresponding to $c$ object classes $C = \{o_1, o_2, ..., o_j, ..., o_c\}$. The class with maximum anchor scores is decided as the label of that RoI-pixel. Let this score be denoted as $RoI_{sc}$. Further, each object in the image contains one or more such RoI-pixels, each having a $RoI_{sc}$. These scores are used to obtain the recognition score and class label of the object. The information about the center co-ordinates, height, width, and class label of the anchor having maximum classification score for each RoI-pixel is used to fit the bounding-box over the object. This is explained in the next step.

*Step 2:* Fitting bounding-box.

All the anchors that contain maximum classification scores for all RoI-pixels are used to fit the bounding-box(s) over the object(s). A RoI-pixel (hence an anchor) may belong to more than one object in case of occlusion. Therefore, we compute the degree of overlapping ($\alpha$) among the anchors considering two together, to check if they are from the same object or from two different objects. $\alpha$ of two anchors $a^1$ and $a^2$ is defined by the cardinality of their intersection, normalized by that of their union. If the class label of two anchors are same and $\alpha$ exceeds a threshold, say $\alpha_T$(usually taken $0.5$), then, anchors $a^1$ and $a^2$ are decided to be of the same object; otherwise, from two occluded objects. If $\alpha = 0$ (in case of non-overlapping), $a^1$ and $a^2$ are from two separate objects. Anchors, thus decided to represent the same object, are combined (union) to form the bounding-box of the object. As said earlier, spatio-temporal granules (refer to the step 2 in Section II-A1) represent objects region [15] in the image/video frame. Therefore, the common region between each anchor and corresponding spatio-temporal granulated region is used as the bounding-box over each occluded/separated object. In the next step, the bounding-box information, such as class labels and $RoI_{sc}$ scores are used to check whether the box belongs to either object or background. Then, the recognition score and class label of the detected object are obtained.

*Step 3:* Object recognition.

Let a bounding-box be characterized by $\zeta$ number of RoI-pixels. Then, it would have $\zeta$ number of $RoI_{sc}$ values with class labels $(RoI_{sc}, o_k^a)_{a=1,2,...,\zeta}^{k\in c}$. Based on these $(RoI_{sc}, o_k^a)$-information, the class label and recognition score of the object $(o)$ are determined. Following two cases may arise:

*Case 1:* Class labels for all the $\zeta$ number of RoI-pixels are same, say, $o_k$, but with different $RoI_{sc}$-values. Then, all the RoI-pixels are called positive RoI-pixels. The recognition score $S^o$ of the object $(o)$ is then determined by the maximum of these $\zeta$ number of $RoI_{sc}$-values.

*Case 2:* Class labels for all the $\zeta$ number of RoI-pixels are not same. Then, the object $o$ is assigned to the $k^{th}$ class $o_k$, if at least $(\zeta/2 + 1)$ RoI-pixels (i.e., majority) have the same class label. These pixels are called positive RoI-pixels. The recognition score of the object $o$ is determined by the maximum of $RoI_{sc}$-values of those positive RoI-pixels. If this (majority) condition is not satisfied, $o$ is treated as 'background', and all the $\zeta$ pixels are called negative RoI-pixels.

As we see, the detector G-RCNN is based on the proposed G-AlexNet which, in turn, uses a pre-trained AlexNet architecture. Pre-trained AlexNet provides the information on class-label of objects and bounding-boxes around them. Even though the AlexNet was pre-trained, embedding of three layers (i.e., granulation, RoI generation, and anchoring process) within it, necessitates its re-training to change the filter parameters in order to make the G-RCNN functional. The training of G-RCNN is explained in the next section.

### *B. Training of G-RCNN*

For training of G-RCNN, we assign a binary class label (either positive or negative) to an anchor at each location in $f_t^G$ using its $c$ classification probabilities and intersection over union ($IoU$) values with respect to various ground truth bounding-boxes. Let there be $l(= 9)$ number of anchors and $r$ number of ground-truth boxes corresponding to each location in $f_t^G$. We assign positive label to an anchor, if its $IoU > 0.75$ [2] for at least one ground-truth box. The anchor is negative, if its $IoU < 0.25$ [2] with respect to all $r$ ground-truth boxes. Only positive and negative anchors over positive RoIs are considered for the training of G-RCNN. Training using these positive and negative anchors of $f_t^G$ is done using back-propagation algorithm based on the output of FC3, with stochastic gradient descent (SGD) search [2] and loss function (say, $L$). The loss function and parameter initialization for the training of G-RCNN are defined in Sections II-B1 and II-B2.

*1) Loss Function:* Loss function $(L)$ has two components, namely classification loss $L_{cl}$ and bounding-box regression loss $L_{bb}$. Let $a_i^j$ be the $i^{th}$ anchor at $j^{th}$ location in $f_t^G$. Let $v_i^j = \{v_i^j(1), v_i^j(2), ..., v_i^j(c)\}_{i=1}^l$, be the set of $c$ classification scores for $a_i^j$, as provided by $c$ number of SVMs. Each SVM designates a particular class. Out of these $c$ values, we select the $v_i^j$ value whose SVM-class label matches with the ground-truth class-information $\{g^j(k)\}, k \in c$. If $\varpi^{th}$ $(\varpi \in c)$ class is matched, then, the log error associated with the said decision on $a_i^j$ is called classification loss $L_{cl}(a_i^j)$. This is defined as:

$$L_{cl}(a_i^j) = cls(a_i^j, g^j) \times log\left\{1 - v_i^j(\varpi)\right\} \tag{5}$$

where desision indicator $cls(a_i^j, g^j)$ is defined as:

$$cls(a_i^j, g^j) = \left\{ \begin{array}{l} 1, \text{ if } a_i^j(\varpi) = g^j(\varpi) \\ 0, \text{ otherwise} \end{array} \right\} \tag{6}$$

Let $bx_i^j$ be the bounding-box of the anchor $a_i^j$ in $f_t^G$. Let $r^j$ be the ground-truth bounding-box at the $j^{th}$ location. Let $b(.)$ be the vector representing four parameterized values (center coordinates, height and width) of a bounding-box (.). Parameterized (normalized) values [2] are used to compute the bounding-box regression error or loss $L_{bb}$ due to the mismatch between $r^j$ and $bx_i^j$. This is defined as

$$L_{bb}(a_i^j) = obj(bx_i^j) \times R|b(bx_i^j) - b(g^j)| \tag{7}$$

where the decision indicator $obj(bx_i^j)$ is defined as:

$$obj(bx_i^j) = \left\{ \begin{array}{l} 1, \text{ if } a_i^j \text{ is a positive anchor} \\ 0, \text{ otherwise} \end{array} \right\} \quad (8)$$

and $R$ denotes the smoothing error function, defined in [10]. Therefore, the total loss $L$ associated with the aforesaid classificatory decision for an anchor $a_i^j$ in $f_t^G$ is defined as

$$L(a_i^j) = L_{cl}(a_i^j) + L_{bb}(a_i^j) \quad (9)$$

Error based on this loss function (refer to Eq. (9)) is back-propagated from the output of FC3 to Conv1 using all the positive RoIs of $f_t^G$ feature map to obtain the optimal weights of each layer in G-RCNN. Parameter initialization is required for the training of G-AlexNet, as defined in the next section.

*2) Parameter Initialization:* All the new layers (e.g., granulation, RoI generation, and anchoring process) in G-AlexNet are randomly initialized with a Gaussian distribution of zero mean and standard deviation of 0.01. Other layers (i.e., the shared convolutional layers) are initialized by those of the pre-trained AlexNet, which was trained over VOC07 data. The momentum and decay parameters of stochastic gradient descent search algorithm during training of G-AlexNet are set to default values 0.9 and 0.001, respectively.

After detecting the objects in a video frame by G-RCNN, we track them. Detection of objects over frames leads to the generation of several possible trajectories corresponding to each detected object. Determining the exact trajectory for such an object is called tracking. This is done by the proposed MCD-SORT algorithm, as described in Section III.

## III. Proposed MCD-SORT Method for Object Tracking

MCD-SORT is an advanced version of the well-known deep SORT [20] used for tracking. The working principle of deep SORT is stated in Section C: S.I (see the supplementary material S.I). Deep SORT can handle both short and long-term occlusions. It functions as follows: first, all possible assignments (trajectories) are computed between the target object in $f_t$ and all existing track-lets in previous frames. Second, the most suitable assignment for each target object is predicted. Consideration of the track-lets in previous frames for all classes increases the searching space and reduces the tracking speed. MCD-SORT overcomes this issue by restricting the computation of all such possible assignments only between the target object and existing track-lets of the same class.

Let $\{t : t_1, t_2, ..., t_n\}$ be the set of $n$ trajectories containing different numbers of track-lets. Let $\{t_\beta : t_\beta^1, t_\beta^2, ..., t_\beta^\xi\}$ be a set that represents the appearance descriptor of all the track-lets of $\beta^{th}$ trajectory. $\xi$ implies the total number of track-lets present in $\beta^{th}$ trajectory. Let $\{o : o_1, o_2, ..., o_\nu, ..., o_m\}$ be the set of $m$ new objects detected in the current frame, as obtained from G-RCNN. In MCD-SORT, we compute the similarity (with a metric $Q$) to associate the $\nu^{th}$ newly detected object with each trajectory having the same class track-lets, and select the one with maximum similarity. The metric $Q$ between $\beta^{th}$ trajectory ($t_\beta$) and $\nu^{th}$ newly detected object ($o_\nu$) is defined as

$$Q(t_\beta, o_\nu) = \left\{ \begin{array}{l} (\lambda_1)E(t_\beta, o_\nu) + (1 - \lambda_1)A(t_\beta, o_\nu), \text{ if } o_\nu^k = t_\beta^k \\ 0, \text{ otherwise} \end{array} \right\} \quad (10)$$

where $k \in c$, $\beta \in n$, $\nu \in m$, $E(t_\beta, o_\nu)$, and $A(t_\beta, o_\nu)$ are two different distance metrics in 8-dimensional (viz, center co-ordinates, height and aspect ratio of the bounding-box, and their velocities) state space. These metrics are defined so as to take care of both short- and long-term occlusions in tracking of $o_\nu$. Lower the value of $Q$ is, higher is the similarity, which is preferable. The effect of $E$ and $A$ on $Q$ is controlled using $\lambda_1$. Metric $E(t_\beta, o_\nu)$ between $t_\beta$ and $o_\nu$ is:

$$E(t_\beta, o_\nu) = \sqrt{o_\nu - t_\beta^1} \quad (11)$$

where $t_\beta^1$ denotes the predicted 8-dimensional Kalman state of the latest track-let of $\beta^{th}$ trajectory. Metric $A(t_\beta, o_\nu)$, on the other hand, is defined using the minimum cosine distance among the $\xi$ track-lets of $t_\beta$ and $o_\nu$, as

$$A(t_\beta, o_\nu) = \min \left\{ 1 - cos(o_\nu, t_\beta^{k_1}) \right\} \quad (12)$$

where $cos(o_\nu, t_\beta^{k_1})$ represents the cosine similarity between $o_\nu$ and $t_\beta^{k_1}$, and $k_1 = 1, 2, ..., \xi$.

One may note that, while Eq. (12) addresses the association of the $o_\nu$ with the $t_\beta$, Eq. (11) does so only for the latest track-let of $t_\beta$. While computing $Q(t_\beta, o_\nu)$ (refer to Eq. (10)), one may further put some thresholds, say, $T_E$ and $T_A$ on $E$ and $A$ values, respectively, for excluding the unlikely association of the newly detected object; thereby reducing the search space and hence, the computation time. The metric $Q(t_\beta, o_\nu)$ (Eq. (10)) defines the distance between $o_\nu$ and $t_\beta$. Accordingly, for $n$ trajectories, there would be $n$ number of $Q$-values for the $o_\nu$. The one with lowest $Q$-value is decided as the assigned trajectory of the $o_\nu$. That is, the association rule for $o_\nu$ is:

$$D(t_\beta, o_\nu) = \left\{ \begin{array}{l} 1, \text{ if } Q(t_\beta, o_\nu) = \min \{Q(t_\beta, o_\nu) \neq 0\} \\ 0, \text{ otherwise} \end{array} \right\} \quad (13)$$

All the said tracking operations are done for each detected object to get its matched or unmatched association with existing trajectories. If an existing trajectory does not get associated with any newly detected object, then, one null track-let record is added to its history. A trajectory with the number of null track-lets greater than a predefined threshold (say, $T_{max}$), is considered to have left the scene, and therefore, deleted from the trajectory set $\{t\}$. The pseudo-code of the MCD-SORT is depicted in Algorithm 2: S.II (see the supplementary material S.II).

## IV. Experimental Results

In Section II, we have described the proposed model (G-RCNN) by introducing the concept of granulation in deep CNN architecture to generate the RoI map that represents various object regions in images/video; thereby, increasing the detection accuracy. While tracking, the system uses an advanced version of

deep SORT algorithm called MCD-SORT (refer to Section III), for accurate and speedy searching. In this section, we experimentally demonstrate these features extensively over 37 videos and provide a comparative study with several state-of-the-art methods of both detection and tracking. Our experiment has four broad objectives. These are to demonstrate:

1) (a) How G-RCNN improves over Fast RCNN and Faster RCNN; thereby signifying the importance of granulation in RoI map generation.
   (b) How spatio-temporal granules improves object detection over spatial granules.
2) Superiority of G-RCNN over some state-of-the-art video object detection techniques, such as TPN [25], D&T [26], and THP [27].
3) The effectiveness of MCD-SORT over SORT and deep SORT.
4) Superiority of MCD-SORT over some state-of-the-art tracking algorithms, such as SOP [17], AMIR15 [28], and AM [29], given a detection algorithm.

The experimental set up, performance metrics, datasets used, and experimental results are discussed in following sections:

### A. Experimental Setup

Algorithms are coded in Python 3.6 (anaconda), with Ubuntu 14.04 using an Intel $i7$ processor clocked at 2.50 GHz and 8.00 GB of memory. The libraries used are cv2, Numpy, and TensorFlow. The GPU used is Quadro 6500. As mentioned in Section II, the feature generation network of G-RCNN (e.g., G-AlexNet) is developed over a pre-trained AlexNet. Training of G-RCNN and pre-trained AlexNet are based on single scale images of $227 \times 227 \times 3$ pixels corresponding to R, G, and B. Test images are re-scaled accordingly. We have used the default parameters of AlexNet in this experiment. These include, $\vartheta = 96$ filters each having size $11 \times 11$, with stride $= 4$ in the Conv1 layer; a filter of size $3 \times 3$ with stride $= 2$ in Pool1, Pool2 and Pool5 layers; 256 filters each having size $5 \times 5$ with padding $= 2$ in Conv2 layer; 384 filters each having size $3 \times 3$ and padding $= 1$ in Conv3 and Conv4 layers; and 256 filters each having size $3 \times 3$ with padding $= 1$ in Conv5 layer. Anchors used over RoI map in Pool5 layer for object localization are of different sizes and different aspect ratios, e.g.,$10 \times 14$, $14 \times 10$, $14 \times 14$, $20 \times 28$, $28 \times 20$, $28 \times 28$, $40 \times 56$, $56 \times 40$, and $56 \times 56$. Sizes of these anchor maps are selected based on the sizes of objects present in the training images/video frames. For object tracking by MCD-SORT (Section III), two thresholds, $T_A = 9$ and $T_E = 3$ with $\lambda_1 = 0.5$, and $T_{max} = 30$ [20] are used, respectively.

### B. Performance Metrics

The performance metrics used in the study are mAP: mean average precision (%), MOTA: multi object tracking accuracy (%), IDS: identity switches (number)[30] , MOTP: multi object tracking precision (%), MT: mostly tracked targets (%), ML: mostly lost targets (%), and Speed (frames per second (fps)). A detailed description of all these performance metrics is stated in Section A: S.III (see the supplementary material S.III). Metrics mAP and Speed are used for object detection [2], while MOTA, IDS, MOTP, MT, ML, and Speed are used for object tracking [30].

### C. Dataset Created and Used

The effectiveness of our G-RCNN and MCD-SORT algorithms along with various comparisons has been demonstrated extensively over several benchmark datasets. Various static images and videos, containing different number of classes, as used for training and testing are explained in the Sections IV-C1 and IV-C2, respectively. Training performances of G-RCNN and Faster RCNN are demonstrated in Section B: S. III (see the supplementary material S.III).

*1) Static Image Data:*

i) VOC-train [12]: It has two parts, namely VOC 07-train ($07_{trn}$) and VOC 12-train ($12_{trn}$). Both $07_{trn}$ and $12_{trn}$ data contain $5K$ and $5.7K$ train images of twenty object classes
ii) VOC-test [12]:VOC-test has two parts, e.g., VOC 07-test ($07_{tst}$) and VOC 12-test ($12_{tst}$). $07_{tst}$ and $12_{tst}$ contain $5K$ and $5.7K$ test images of twenty object classes.

*2) Videos:*

i) MOT15 challenge dataset [22]: This dataset has 11 videos each containing either one and/or two object classes, namely person and car.
ii) Video analytics road 19 (VAR19): It has two parts, namely video analytics road 19-train ($19_{trn}$) and video analytics road 19-test ($19_{tst}$). $19_{trn}$ data contains $2K$ train video-frames of eight object classes (e.g., car, bus, loco, truck, minivan, bike/scooter, bicycle, and person). Whereas, $19_{tst}$ contains 30 real-world videos with the aforesaid eight different classes of objects.
iii) Urban Tracker: It has two parts, namely Urban Tracker-train ($ut_{trn}$) and Urban Tracker-test ($ut_{tst}$) [23]. Both $ut_{trn}$ and $ut_{tst}$ data contain 626 train video-frames and 4 videos, respectively, each having four different object classes e.g., bicycle, bike/scooter, person, and car.

MOT15 challenge, $19_{tst}$, and $ut_{tst}$ data are used as test videos. From MOT15 challenge, we have used three videos, such as TUD-Crossing (TUD), PETS09-S2L2 (PETS), and AVG-Town Center (AVG). The total number of classes or objects ($C_{test}$) over these test data is eight.

For training, in general, we have used various static images and video data, e.g., $07_{trn}$, $12_{trn}$, $19_{trn}$, and $ut_{trn}$, containing different number of classes ($C_{train}$). Note that four object classes in $ut_{trn}$ and eight object classes in $19_{trn}$ are common in the aforesaid eight test classes ($C_{test}$). Whereas, twenty object-classes in $07_{trn}$ and $12_{trn}$ contain those six test classes and extra fourteen classes. That means $C_{test}$ and $C_{train}$ may not be the same always in the experiment. In some parts of the experiment, for the sake of fair comparison with Fast RCNN and Faster RCNN, we have used $07_{tst}$ and $12_{tst}$ data.

Out of these data sets, $19_{trn}$ and $ut_{trn}$ are created in our lab based on real-life videos acquired from a steel plant in India and urban tracker videos. VAR19 is a new real-life traffic data, not available elsewhere, unlike VOC and MOT 15 which are

TABLE I
COMPARATIVE PERFORMANCE OF G-RCNN-RPN IN TERMS OF MAP (%)

| Detector | Architecture | Training data | Test data | mAP |
|---|---|---|---|---|
| Fast RCNN | AlexNet | $07_{trn}$ | $07_{tst}$ | 57.1 |
| Fast RCNN | AlexNet | $07_{trn} + 12_{trn}$ | $07_{tst}$ | 60.9 |
| Fast RCNN | AlexNet | $07_{trn} + 12_{trn}$ | $12_{tst}$ | 61.3 |
| Faster RCNN | AlexNet | $07_{trn}$ | $07_{tst}$ | 65.2 |
| Faster RCNN | AlexNet | $07_{trn} + 12_{trn}$ | $12_{tst}$ | 63.7 |
| G-RCNN-rpn | G-AlexNet | $07_{trn}$ | $07_{tst}$ | 79.4 |
| G-RCNN-rpn | G-AlexNet | $07_{trn} + 12_{trn}$ | $07_{tst}$ | 80.1 |
| G-RCNN-rpn | G-AlexNet | $07_{trn} + 12_{trn}$ | $12_{tst}$ | 80.9 |

TABLE II
PERFORMANCE OF G-RCNN-FRPN AND G-RCNN-RPN IN TERMS OF MAP (%)

| Detector | Training data | Test data | mAP |
|---|---|---|---|
| G-RCNN-rpn | $07_{trn} + 12_{trn} + 19_{trn}$ | $19_{tst}$ | 77.4 |
| G-RCNN-rpn | $07_{trn} + 12_{trn} + ut_{trn}$ | $ut_{tst}$ | 74.3 |
| G-RCNN-rpn | $07_{trn} + 12_{trn} + 19_{trn} + ut_{trn}$ | $19_{tst}$ | 78.0 |
| G-RCNN-rpn | $07_{trn} + 12_{trn} + 19_{trn} + ut_{trn}$ | $ut_{tst}$ | 75.2 |
| G-RCNN-frpn | $07_{trn} + 12_{trn} + 19_{trn}$ | $19_{tst}$ | 80.2 |
| G-RCNN-frpn | $07_{trn} + 12_{trn} + ut_{trn}$ | $ut_{tst}$ | 77.6 |
| G-RCNN-frpn | $07_{trn} + 12_{trn} + 19_{trn} + ut_{trn}$ | $ut_{tst}$ | 77.6 |
| G-RCNN-frpn | $07_{trn} + 12_{trn} + 19_{trn} + ut_{trn}$ | $19_{tst}$ | 80.6 |

benchmark data available online. For annotation of the $19_{trn}$ and $ut_{trn}$ video-frames, we have labelled all the objects present, fully or by more than 80%, in a frame by their respective categories. Bounding-boxes over them are marked, and their co-ordinates are recorded. If an object is occluded by less than 20%, then, it is marked as a 'truncated' object of the concerned category.

### D. Results on Object Detection and Tracking

Experiments along with comparisons are conducted to evaluate the effectiveness of the proposed algorithms (G-RCNN + MCD-SORT) in multiple object detection and tracking, in line with the four objectives (stated in the beginning of Section IV). The objective $1(a)$ does not require G-RCNN to consider the temporal information of images, while the objective $1(b)$ does. That is, $1(a)$ and $1(b)$ need static images and videos, respectively as inputs. Accordingly, G-RCNN used in objectives $1(a)$ and $1(b)$ may be designated as G-RCNN-rpn and G-RCNN-frpn, respectively. Objectives $2, 3$, and 4 need videos as inputs for object detection and tracking. Objectives $1(a)$, $1(b)$, and 2 are discussed in Sections IV-D1, IV-D2, and IV-D3. Whereas, objectives 3 and 4 are demonstrated in Section IV-D4.

*1) Significance of Granulation in RoI Proposal:* To demonstrate the significance of granulation in RoI proposal, we provide comparative results for object detection of the proposed G-RCNN-rpn with Fast RCNN and Faster RCNN in Table I for $07_{tst}$ and $12_{tst}$ data. As said earlier, G-RCNN is developed over AlexNet architecture. Therefore, Fast RCNN and Faster RCNN are developed over the same architecture to make a fair comparison. As $07_{tst}$ and $12_{tst}$ data are used for testing purpose, we have used either $07_{trn}$ or the combination of $07_{trn}$ and $12_{trn}$ for training. First five rows (refer to Table I) correspond to results with Fast RCNN and Faster RCNN for different combinations of training and testing sets. The remaining are for G-RCNN-rpn with same sets of combinations. It is seen from the table that incorporating granulation in deep CNN (viz, G-RCNN-rpn) results in superior mAP score over all the respective cases. This signifies that the RoI-pixels detected by the proposed G-RCNN-rpn are more representative of object classes. GPU Quadro 6500 is used for all these detectors. The speed of Fast RCNN, Faster RCNN, and G-RCNN algorithms are 1.4 fps, 6 fps, and 5.5 fps, respectively. Therefore, G-RCNN-rpn is also superior to Fast RCNN in terms of speed (fps); whereas, Faster RCNN is little faster.

*2) Significance of Spatio-Temporal Information in RoI Proposal:* Here, we demonstrate the relevance of using the spatio-temporal information (in G-RCNN-frpn) over the spatial information (in G-RCNN-rpn) for the task of granulation. Therefore, one needs video data for temporal information. Accordingly, in this experiment, G-AlexNet is retrained with video data like $19_{trn}$ and $ut_{trn}$ on the top of $07_{trn}$ and $12_{trn}$, and tested over $19_{tst}$ and $ut_{tst}$ videos. Table II shows the comparative performance in terms of mAP for different combinations of training data. First four rows correspond to the results of G-RCNN-rpn and the remaining are for the G-RCNN-frpn. For all the cases, G-RCNN-frpn is seen to be superior to G-RCNN-rpn in terms of mAP. The speed of G-RCNN-rpn and G-RCNN-frpn are 5.5 fps and 5.7 fps, respectively. Comparative results signify the relevance of spatio-temporal information over spatial information. As expected, considering all the datasets (viz, $07_{trn} + 12_{trn} + 19_{trn} + ut_{trn}$) for training, provides better mAP-score (e.g., row 5 over 1, row 6 over 2, row 7 over 4, and row 8 over 3). These results demonstrate the superiority of G-RCNN-frpn over G-RCNN-rpn. Example results of such detection by G-RCNN-frpn and G-RCNN-rpn for one frame of $19_{tst}$ video are explained in Section C: S.III (see the supplementary material S.III).

*3) Comparison of G-RCNN With State-of-the-Art Video Object Detection Algorithms:* We provide a comparative performance of G-RCNN with some recently developed video object detection techniques, namely TPN [25], D&T [26], and THP [27]. All these techniques are based on deep CNN. We have used $07_{trn}$ (image) and TUD video for training and testing, respectively. One may note that the aforesaid comparing techniques use different feature generation networks (e.g., GoogLeNet and ResNet 101), and different GPUs, like Titan $X$ and $K40$. Our G-RCNN uses G-AlexNet as feature generation network and Quadro-6500 as GPU. Detection accuracy (mAP) depends on feature generation network only. Whereas, detection speed involves both the computation time of feature generation network and computing power of GPUs. Therefore, it may appropriate and fair to make the comparison only in terms of mAP that would reflect the effectiveness of the models with respect to their feature generation networks. The results are reported in Table III. It is seen that the G-RCNN is the best among the three comparing detector in terms of mAP score.

*4) Tracking Results:* Here, we discuss, as per objectives 3 and 4 (beginning of Section IV), the tracking performance of the proposed MCD-SORT, given any object detector. We have

TABLE III
PERFORMANCE IN TERMS OF MAP (%), AND SPEED (FPS) FOR VARIOUS VIDEO OBJECT DETECTORS

| Detector | Architecture | mAP | Speed | GPU |
|---|---|---|---|---|
| TPN [25] | GoogLeNet | 69.1 | 2.1 | Titan X |
| D & T [26] | ResNet 101 | 76.3 | 7.8 | Titan X |
| THP [27] | ResNet 101 | 78.4 | 8.6 | K40 |
| G-RCNN | G-AlexNet | 80.6 | 5.7 | Quadro 6500 |

TABLE IV
TRACKING RESULTS OF SORT, DEEP SORT, AND MCD-SORT OVER $19_{tst}$, $ut_{tst}$, TUD, AND PETS VIDEOS

| Detector+Tracker | Data | MOTA | MOTP | MT | ML | IDS |
|---|---|---|---|---|---|---|
| Faster RCNN+SORT | $19_{tst}$ | 56.6 | 76.2 | 25.4 | 22.7 | 3172 |
| Faster RCNN+deep SORT | $19_{tst}$ | 64.4 | 77.1 | 32.8 | 18.2 | 2953 |
| Faster RCNN+MCD-SORT | $19_{tst}$ | 65.3 | 77.4 | 33.5 | 17.1 | 2841 |
| G-RCNN+SORT | $19_{tst}$ | 60.8 | 78.3 | 30.1 | 18.4 | 2810 |
| G-RCNN+deep SORT | $19_{tst}$ | 66.3 | 78.9 | 38.2 | 15.7 | 1945 |
| G-RCNN+MCD-SORT | $19_{tst}$ | 69.1 | 79.3 | 45.7 | 12.8 | 859 |
| Faster RCNN+SORT | $ut_{tst}$ | 52.1 | 70.6 | 25.2 | 23.9 | 305 |
| Faster RCNN+deep SORT | $ut_{tst}$ | 58.3 | 72.1 | 31.6 | 21.6 | 269 |
| Faster RCNN+MCD-SORT | $ut_{tst}$ | 59.2 | 72.1 | 33.4 | 18.2 | 215 |
| G-RCNN+SORT | $ut_{tst}$ | 55.7 | 72.8 | 32.9 | 19.1 | 221 |
| G-RCNN+deep SORT | $ut_{tst}$ | 61.5 | 74.1 | 36.8 | 14.2 | 192 |
| G-RCNN+MCD-SORT | $ut_{tst}$ | 62.4 | 74.7 | 38.2 | 13.8 | 173 |
| Faster RCNN+SORT | TUD | 67.5 | 74.5 | 46.2 | 7.7 | 18 |
| Faster RCNN+deep SORT | TUD | 69.9 | 74.2 | 51.4 | 4.1 | 15 |
| Faster RCNN+MCD-SORT | TUD | 72.5 | 78.1 | 53.6 | 3.7 | 6 |
| G-RCNN+SORT | TUD | 77.2 | 76.3 | 51.9 | 4.2 | 4 |
| G-RCNN+deep SORT | TUD | 78.6 | 79.2 | 54.3 | 3.9 | 4 |
| G-RCNN+MCD-SORT | TUD | 80.1 | 80.9 | 61.8 | 3.6 | 3 |
| Faster RCNN+SORT | PETS | 46.1 | 72.2 | 16.7 | 4.8 | 193 |
| Faster RCNN+deep SORT | PETS | 58.7 | 80.2 | 18.3 | 4.1 | 182 |
| Faster RCNN+MCD-SORT | PETS | 58.7 | 80.2 | 18.3 | 4.1 | 182 |
| G-RCNN+SORT | PETS | 59.3 | 80.7 | 18.6 | 3.9 | 178 |
| G-RCNN+deep SORT | PETS | 60.1 | 81.9 | 20.3 | 2.6 | 175 |
| G-RCNN+MCD-SORT | PETS | 60.1 | 81.9 | 20.3 | 2.6 | 175 |

TABLE V
COMPARING TRACKING PERFORMACE OF MCD-SORT OVER TUD AND AVG VIDEOS

| Detector+Tracker | Data | MOTA | MOTP | MT | ML | IDS |
|---|---|---|---|---|---|---|
| Faster RCNN+SOP | TUD | 46.2 | 61.8 | 9.3 | 18.6 | 64 |
| Faster RCNN+AMIR15 | TUD | 60.3 | 66.9 | 38.7 | 12.1 | 37 |
| Faster RCNN+AM | TUD | 52.4 | 66.1 | 36.2 | 21.6 | 15 |
| Faster RCNN+MCD-SORT | TUD | 72.5 | 78.1 | 53.6 | 3.7 | 6 |
| G-RCNN+SOP | TUD | 51.8 | 64.2 | 10.6 | 15.0 | 51 |
| G-RCNN+AMIR15 | TUD | 62.1 | 69.7 | 41.1 | 11.3 | 23 |
| G-RCNN+AM | TUD | 56.2 | 68.3 | 38.0 | 21.1 | 14 |
| G-RCNN+MCD-SORT | TUD | 80.1 | 80.9 | 61.8 | 3.6 | 3 |
| Faster RCNN+SOP | AVG | 8.9 | 69.6 | 1.3 | 68.1 | 109 |
| Faster RCNN+AMIR15 | AVG | 36.2 | 69.5 | 26.1 | 17.7 | 234 |
| Faster RCNN+AM | AVG | 37.5 | 68.1 | 14.2 | 30.5 | 79 |
| Faster RCNN+MCD-SORT | AVG | 39.7 | 71.1 | 32.8 | 12.5 | 74 |
| G-RCNN+SOP | AVG | 9.2 | 70.1 | 2.4 | 66.9 | 97 |
| G-RCNN+AMIR15 | AVG | 36.4 | 68.1 | 26.7 | 16.5 | 226 |
| G-RCNN+AM | AVG | 37.9 | 70.2 | 18.1 | 27.4 | 71 |
| G-RCNN+MCD-SORT | AVG | 57.5 | 73.3 | 46.1 | 10.2 | 51 |

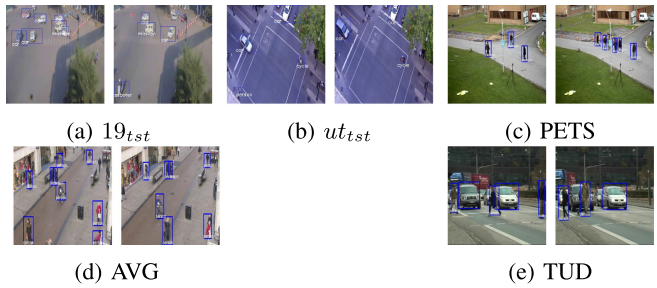


Fig. 2. Tracking results of MCD-SORT with G-RCNN.

demonstrated that MCD-SORT is not only an advanced version of SORT [18] and deep SORT [20], but also superior to some state-of-the-art trackers, such as SOP [17], AMIR15 [28], and AM [29]. In this study, 30 videos from $19_{tst}$, 4 videos from $ut_{tst}$, and 3 videos (e.g., TUD, PETS, and AVG) from MOT15 challenge are used. $19_{tst}$, and $ut_{tst}$ have multi-class objects. TUD contains objects of two classes. PETS and AVG have single class (person) objects only.

Table IV shows the performance of MCD-SORT, deep-SORT, and SORT for tracking over the datasets $19_{tst}$, $ut_{tst}$, TUD, and PETS. From Table IV (first 18 rows), it is evident that MCD-SORT is superior to SORT and deep SORT for multi-class and two-class tracking in terms of all the metrics, respectively. The speed of SORT and deep SORT algorithms are 60 fps and 21 fps, respectively with the GPU TitanX. Whereas, the speed of MCD-SORT algorithm is 29fps with the GPU Quadro 6500. As per speed, SORT has an edge since it uses only motion information instead of both motion and appearance used in deep SORT and MCD-SORT. For single-class tracking, as mentioned in Section III, MCD-SORT boils down to deep SORT. This is evident in Table IV (refer to rows 20 and 21, and rows 23 and 24). Additional results on the performance of MCD-SORT, deep SORT, and SORT over the dataset AVG are explained in Section D: S.III (refer to the supplementary material S.III).

For comparing the tracking performance of MCD-SORT with the recently developed trackers, e.g., SOP, AMIR15, and AM, we provide results in Table V corresponding to TUD and AVG videos. The tracking results shown corresponding to Faster RCNN and G-RCNN as detectors. The speed of these tracking algorithms, such as SOP, AMIR15, and AM, are 14 fps, 6 fps, and 11 fps, respectively. From Table V (last 8 rows), dealing with single-class object(s) tracking in AVG video, it is evident that MCD-SORT is the best among all the comparing state-of-the-art trackers with respect to all performance metrics for both detectors. Moreover, the tracking performance (indices-wise) is seen to be much better with the detector G-RCNN than Faster RCNN. This further supports our claims in Section IV-D1 regarding the superiority of G-RCNN over Faster RCNN. Additional results on the performance of MCD-SORT, SOP, AMIR15, and AM over the dataset PETS are shown in Section D: S.III (see the supplementary material S.III).

Similar conclusions also hold good for tracking two-class objects (first 8 rows from Table V). For tracking with two-class objects (persons and cars) in TUD video, it is seen that MCD-SORT performs the best compared to the comparing trackers in terms of all the metrics corresponding to both detectors G-RCNN and Faster RCNN. Note that since the investigation involves two-class tracking, the comparing methods (viz, SOP, AMIR15, and AM) which were originally designed for single-class tracking, are retrained here for two-class problem for the sake of fair comparison. Further, among all the combinations of detectors and trackers in Tables IV to V, proposed G-RCNN and MCD-SORT results in the best tracking performance. Example

results of such tracking by G-RCNN+MCD-SORT for $19_{tst}$, $ut_{tst}$, PETS, AVG, and TUD videos are shown in Fig. 2.

## V. Conclusion

An object detector called granulated RCNN (G-RCNN) is described based on a newly developed deep CNN architecture called G-AlexNet. G-AlexNet is an advanced version of AlexNet where the concept of granulation is incorporated in the output map of its Pool1 layer. G-AlexNet acts as the feature generation network of G-RCNN to produce the effective regions of interest (RoIs) of objects in videos for their detection, in unsupervised mode. Both spatio-color neighborhood and temporal granules are formed in this process. Unlike Fast RCNN [10] and Faster RCNN [2], G-RCNN accepts videos directly as inputs. During the training of G-RCNN, the loss function is based only on the positive and negative anchors over the positive RoIs, unlike Fast RCNN and Faster RCNN where all RoIs are used. The said RoIs that are generated by G-RCNN, characterize the object(s) regions more accurately as compared to Fast RCNN and Faster RCNN. In G-RCNN, RoIs are generated using the granulation over the pooling feature map. Whereas, in Fast RCNN and Faster RCNN, the entire pooling features are used as RoIs. The task of classification is restricted only to the objects in RoIs, rather than considering the entire feature map, which leads to a significant improvement in on-line object-detection accuracy of G-RCNN. The use of spatio-temporal information in forming foreground granules, as expected, is more effective than using spatial information alone. The concept of using granulation in deep CNN is unique. In another part, a new tracking algorithm, namely multi-class deep SORT (MCD-SORT) is designed which is an advanced version of the popular deep SORT where the computation of association of detected objects with the existing trajectories is restricted only within the objects of same category. This reduces the runtime while increasing the tracking accuracy.

Object detector G-RCNN is not only an improved version of the well-known Fast RCNN and Faster RCNN, but also found superior to the recently developed video object detectors TPN [25], D&T [26], and THP [27] when tested over TUD-crossing video (TUD). For a given detector, the proposed tracker MCD-SORT, an advanced version of deep SORT, outperforms some state-of-the-art trackers, e.g., SOP [17], AMIR15 [28], and AM [29] for tracking single-class and two-class objects. Among all the combinations of detectors and trackers considered in the experiment, the one like G-RCNN + MCD-SORT is found to be the best. We have used here the trained AlexNet as the basis to develop the G-AlexNet, the proposed feature generation network of G-RCNN. However, one can use any other type of deep CNN to model the G-RCNN. VAR 19 data is prepared by the authors based on the real-life videos acquired from a Steel Plant in India. The preparation of VAR 19 is significant, particularly, when the availability of real-life video data is scarce.

Although the proposed G-RCNN has been demonstrated for obejct detection from traffic images, it can be made applicable for other kind of images, such as SAR iamge. In this context, recent application of deep CNN for change detection [31] in SAR images may be explore as a future study of the proposed G-RCNN, among other applications.

## Acknowledgment

S.K. Pal acknowledges National Science Chair, SERB-DST, Govt. of India, and Indian National Science Academy (INSA) Distinguished Professorship. Authors also acknowledge Safety Analytics and Virtual Reality Laboratory (www.savr.iitkgp.ac.in), Department of Industrial and Systems Engineering, and the Centre of Safety Engineering and Analytics (CoE-SEA), IIT Kharagpur for research facilities for this work. The support provided by the management of the Steel Company is gratefully acknowledged.

## References

[1] S. A. Ahmed, D. P. Dogra, S. Kar, and P. P. Roy, "Trajectory-based surveillance analysis: A survey," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 29, no. 7, pp. 1985–1997, Jul. 2019.

[2] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 91–99.

[3] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 779–788.

[4] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *Int. J. Comput. Vis.*, vol. 104, no. 2, pp. 154–171, 2013.

[5] Z. Tirandaz, G. Akbarizadeh, and H. Kaabi, "Polsar image segmentation based on feature extraction and data compression using weighted neighborhood filter bank and hidden markov random field-expectation maximization," *Measurement*, vol. 153, Mar. 2020, Art. no. 107432.

[6] B. Alexe, T. Deselaers, and V. Ferrari, "Measuring the objectness of image windows," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 11, pp. 2189–2202, Nov. 2012.

[7] G. Akbarizadeh, "A new statistical-based kurtosis wavelet energy feature for texture recognition of sar images," *IEEE Trans. Geosci. Remote Sens.*, vol. 50, no. 11, pp. 4358–4368, Nov. 2012.

[8] L. Jiao *et al.*, "A survey of deep learning-based object detection," *IEEE Access*, vol. 7, pp. 128 837–128 868, 2019.

[9] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2014, pp. 580–587.

[10] R. Girshick, "Fast R-CNN," in *Proc. 2015 Int. Conf. Comp. Vis.*, Santiago, Chile, Dec. 7–13, 2015, pp. 13–16.

[11] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 248–255.

[12] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *Int. J. Comput. Vis*, vol. 88, no. 2, pp. 303–338, 2010.

[13] F. Sharifzadeh, G. Akbarizadeh, and Y. S. Kavian, "Ship classification in sar images using a new hybrid cnn–mlp classifier," *J. Indian Soc. Remote Sens.*, vol. 47, no. 4, pp. 551–562, 2019.

[14] D. B. Chakraborty and S. K. Pal, "Neighborhood rough filter and intuitionistic entropy in unsupervised tracking," *IEEE Trans. Fuzzy Syst.*, vol. 26, no. 4, pp. 2188–2200, Aug. 2018.

[15] D. Chakraborty, B. U. Shankar, and S. K. Pal, "Granulation, rough entropy and spatiotemporal moving object detection," *Appl. Soft Comput.*, vol. 13, no. 9, pp. 4001–4009, 2013.

[16] S. Zhang, X. Lan, H. Yao, H. Zhou, D. Tao, and X. Li, "A biologically inspired appearance model for robust visual tracking," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 10, pp. 2357–2370, Oct. 2017.

[17] J. Shen, Z. Liang, J. Liu, H. Sun, L. Shao, and D. Tao, "Multiobject tracking by submodular optimization," *IEEE Trans. Cybern.*, vol. 49, no. 6, pp. 1990–2001, Jun. 2019.

[18] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *Proc. IEEE Int. Conf. Image Process.*, 2016, pp. 3464–3468.

[19] S. Sun, N. Akhtar, H. Song, A. S. Mian, and M. Shah, "Deep affinity network for multiple object tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, to be published.
[20] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *Proc. IEEE Int. Conf. Image Process.*, 2017, pp. 3645–3649.
[21] S.-H. Bae and K.-J. Yoon, "Confidence-based data association and discriminative deep appearance learning for robust online multi-object tracking," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 3, pp. 595–610, Mar. 2018.
[22] L. Leal-Taixé, A. Milan, I. Reid, S. Roth, and K. Schindler, "Motchallenge 2015: Towards a benchmark for multi-target tracking," *CoRR*, vol. abs, pp. 1–15, 2015.
[23] J.-P. Jodoin, G.-A. Bilodeau, and N. Saunier, "Urban tracker: Multiple object tracking in urban mixed traffic," in *Proc. IEEE Winter Conf. Appl. Comput. Vis.*, 2014, pp. 885–892.
[24] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proc. 27th Int. Conf. Mach. Learn.*, 2010, pp. 807–814.
[25] K. Kang *et al.*, "Object detection in videos with tubelet proposal networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 727–735.
[26] C. Feichtenhofer, A. Pinz, and A. Zisserman, "Detect to track and track to detect," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 3038–3046.
[27] X. Zhu, J. Dai, L. Yuan, and Y. Wei, "Towards high performance video object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 7210–7218.
[28] Q. Chu, W. Ouyang, H. Li, X. Wang, B. Liu, and N. Yu, "Online multi-object tracking using cnn-based single object tracker with spatial-temporal attention mechanism," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 4836–4845.
[29] A. Sadeghian, A. Alahi, and S. Savarese, "Tracking the untrackable: Learning to track multiple cues with long-term dependencies," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 300–311.
[30] K. Bernardin and R. Stiefelhagen, "Evaluating multiple object tracking performance: The clear mot metrics," *J. Image Video Process.*, vol. 2008, pp. 1–10, 2008.
[31] F. Samadi, G. Akbarizadeh, and H. Kaabi, "Change detection in sar images using deep belief network: A new training approach based on morphological images," *IET Image Process.*, vol. 13, no. 12, pp. 2255–2264, 2019.