IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON REAL-TIME MACHINE LEARNING APPLICATIONS IN MOBILE ROBOTICS

Received September 23, 2020, accepted October 5, 2020, date of publication October 26, 2020, date of current version November 12, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3033537

AMMDAS: Multi-Modular Generative Masks Processing Architecture With Adaptive Wide Field-of-View Modeling Strategy

VENKATA SUBBAIAH DESANAMUKULA[1], PREMITH KUMAR CHILUKURI[1], PUSHKAL PADALA[2], PREETHI PADALA[3], AND PRASAD REDDY PVGD[1]

[1]CS&SE, Andhra University College of Engineering (A), Visakhapatnam 530003, India
[2]CSE, The National Institute of Engineering, Mysuru 570008, India
[3]CSE, National Institute of Technology Karnataka, Mangaluru 575025, India

Corresponding author: Premith Kumar Chilukuri (premithchilukuri@gmail.com)

ABSTRACT The usage of transportation systems is inevitable; any assistance module which can catalyze the flow involved in transportation systems, parallelly improving the reliability of processes involved is a boon for day-to-day human lives. This paper introduces a novel, cost-effective, and highly responsive Post-active Driving Assistance System, which is "Adaptive-Mask-Modelling Driving Assistance System" with intuitive wide field-of-view modeling architecture. The proposed system is a vision-based approach, which processes a panoramic-front view (stitched from temporal synchronous left, right stereo camera feed) & simple monocular-rear view to generate robust & reliable proximity triggers along with co-relative navigation suggestions. The proposed system generates robust objects, adaptive field-of-view masks using FRCNN+Resnet-101_FPN, DSED neural-networks, and are later processed and mutually analyzed at respective stages to trigger proximity alerts and frame reliable navigation suggestions. The proposed DSED network is an Encoder-Decoder-Convolutional-Neural-Network to estimate lane-offset parameters which are responsible for adaptive modeling of field-of-view range (157^0-210^0) during live inference. Proposed stages, deep-neural-networks, and implemented algorithms, modules are state-of-the-art and achieved outstanding performance with minimal loss(L{p, t}, L_δ, L_{Total}) values during benchmarking analysis on our custom-built, KITTI, MS-COCO, Pascal-VOC, Make-3D datasets. The proposed assistance-system is tested on our custom-built, multiple public datasets to generalize its reliability and robustness under multiple wild conditions, input traffic scenarios & locations.

INDEX TERMS Adaptive field of view modeling, automotive applications, driving assistance systems, lane detection and analysis, object detection and tracking, spatial auto-correlation.

I. INTRODUCTION

Transportation plays an indispensable role in individual and social welfare, the economy, and quality of life. Society pays monetary (purchase, Functional, and maintenance) costs, social and environmental costs (noise pollution and traffic jams), penalties on detrimental traffic accidents, etc.

Recent studies from the World Health Organization indicate that 1.25 million deaths occur every year due to road traffic accidents.[1] Moreover, such accidents resulted in a global cost of ∼US$518 billion per year, which results in a decline of ∼1-2% gross domestic product from all of the countries in the world[2] [1]. Dynamic visual environmental analysis and understanding are key [2] requirements for any autonomous-mobile systems. The tracking and recognition of traffic participants (pedestrians, cars, bicyclists [3]–[7], etc) which are in proximity plays a crucial role in the safe maneuvering of self-governed autonomous [8] vehicles. Sensors, such as radar and LIDAR [9]–[11], have also been used for detection & tracking purposes, using sensor data fusion [9], [10] approaches. An obstacle, lane detection [12] & tracking with Field of view spatial analysis are key

The associate editor coordinating the review of this manuscript and approving it for publication was Aysegul Ucar.

[1]Refer to "https://www.who.int/violence_injury_prevention/road_safety_status/2015/en/"

[2]Refer to "https://www.asirt.org/safe-travel/road-safety-facts/"

functions of advanced driver-assistance systems (ADAS [8]). In the context of driver-assistance systems [2], the purpose of obstacle-detection and tracking systems [3], [9], [13]–[15] is to detect and monitor & analyze the dynamic-behavior of one or more obstacles in the proximity of the host vehicle. The role of lane detection [12] and road spatial analysis [14], [16], [17], is to analyze and make decisions based on dynamic surroundings for safer navigation. These modules play a predominant role to dynamically process the DAS systems [2], [8] into the surrounding environment. The objective of driving-monitoring and assistance systems (DMAS [2]) is to keep an observation on a driver's driving-status and to provide needful assistance for safe and reliable driving. Such systems help drivers by reinforcing sensing power, providing inputs to aid better decision-making. Depending on their features & functionalities, there are numerous names for autonomous systems such as intelligent-vehicle control systems, advanced driver-assistance systems [8], collision avoidance systems [10], [11], driver's inattention monitoring systems, etc. Therefore, the usage of ADAS [8], DMAS, DAS [2] can aid to avoid potential crashes and to provide the required information for decision-making. Reliable modules for detection and tracking in general ADAS [8], DAS for real-time situations in wild environments have been a challenge from the last 2 decades, predominantly recognizing various classes of objects, their spatio-temporal variance, and their state's approximation from noisy observations & findings at discrete intervals of time. In some cases driver's voluntary behaviors, such as eating, drinking, smoking, or answering a call, etc, are reasons for accidents and road-crashes [1]; drivers may be less likely to engage in these types of behaviors when driving task demands are high, for example, when negotiating a busy intersection, driving in poor visibility, or on a busy multi-lane roadway [18]. From the above discussed factors and reasons safety in automobile systems has been a vital concern since the early days of on-road vehicles to avoid them. Several original-equipment-manufacturers (OEMs) have attempted to solve this issue by developing numerous safety systems [2], [8] (ADAS, DMAS, DAS, accident alerting systems, etc.) to safeguard the occupants simultaneously avoiding collateral damage to external surroundings of the vehicle. Generally, these systems are mainly classified into two types: i) reactive and ii) proactive. Passive safety systems protect vehicle occupants from injuries after a crash, e.g., seat belts, airbags & accident alert systems & fatigue prevention systems, etc. Active systems include lane-keeping and automatic braking [6], [7], [19], Proximity alert [10], [11] systems etc.

In recent years, many vision-only based methodologies have been proposed for their versatility in usage, information they can provide, and their cost efficiency. In these methods, some learning-based approaches only focus on the tracking and detection of specific obstacles [3], [4], [11] (pedestrians and vehicles) and to keep a track on their respective surroundings within the field of view; while motion based methods can detect objects which are in motion [5] and analyze their optical flows [3], [9], [10], [13] in real-time environments. Modern-day robust ADAS are key and under-structure technologies in building reliable autonomous vehicles [8], but several challenges with the design, implementation, and functioning of ADAS and DAS remain yet to be overcome [2], [8]. Some of these challenges include optimizing energy utilization, Cost efficiency, achieving minimal response latency, dynamic adjustment based on weather conditions, and security of the overall system. Moreover, current DAS/assistance systems pertain only for proximity alerting (i.e tracking and detecting objects on-road and alerting the user in case of proximity). So in this paper, the proposed DAS solves the challenges faced (mentioned in section 2, Fig. 25) by current driving assistant systems, by introducing an ''adaptive mask modeling strategy'', where the generated adaptive masks are modeled based on the external dynamic field of view and surrounding traffic flow. The proposed system(AMMDAS) is purely a vision-based approach and is completely built on real-time computational modules using deep neural networks & computer vision techniques. This proposed system eliminates the usage of extra sensors, and considers input only through stereo camera feed (front left-right $\sim 165^0$ & rear $\sim 65^0$ of vehicle and the overall FOV covered by our input feed is $\sim 230^0$), thereby reducing cost, processing power, and response latency. The proposed system follows a multi-stage methodology where each stage is individually built and, the built modules are concurrently executed according to Fig. 3 proposed pipeline to generate and process adaptive masks during inference time to output final proximity alerts along with corresponding robust navigation suggestions to the end-user.

Each stage implemented in this paper plays a significant role in generating end-user's navigation suggestions (Fig. 2) by stitching a panoramic image (Fig. 1(b)), detecting objects in the feed (Fig. 7), adaptively modifying FOVs, and generating l, c, r-masks (Fig. 12), processing intermediate results + calculated parameters and framing proximity alerts along with corresponding logical suggestions, and generation of dense depth maps (Fig. 15). Here adaptiveness refers to the capability of the currently proposed system to dynamically change and adjust the reach of FOV according to the external environment[3] of the host. We initially construct a 165^0 wide panoramic view from input stereo feed, so that any decision & alerts made will be based on a complete broad view understanding of scenes in their surroundings (Fig. 1). This front view's constructed panoramic video frames and concurrent frames from the rear view's video feed are simultaneously passed to an OMM module and a D.F.O.V.M module to generate object masks (Fig. 7) and adaptive left, center, right-field of view (FOV) masks (Fig. 12) based on surrounding objects, vehicles, lanes respectively for each view. These generated

[3]Environment in this proposed paper refers to the external situations present outside of the host vehicle, here external situations refers to the conditions and orientations of live on-road lanes, objects/vehicles and other necessary factors which should be considered to ensure a safer navigation to the end-user.

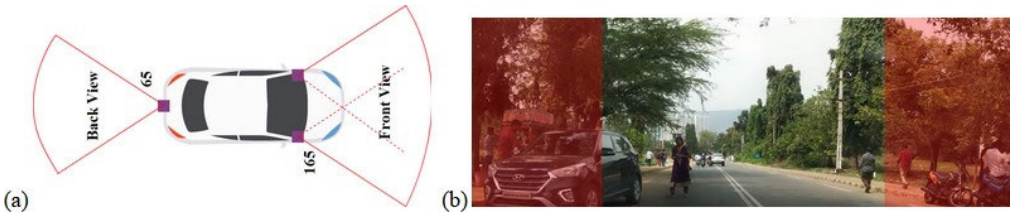

FIGURE 1. (a) Red lines indicate the FOV range outliers covered by our module. Front view covers ∼165^o by stitching a panoramic image from stereo input, and rear-view covers ∼65^0, so the overall FOV coverage is around 230^0. (b) represents the entire 165^0 field of view covered by our proposed system; and the regions highlighted in red are additional fields of view covered by our module due to panoramic stitching, the highlighted FOVs are generally omitted by any DAS method when they input feed via regular mono or stereo visions; this case happens with majority of the existing DAS systems, which might also correspondingly increase the DAS's sensitivity towards FP and FN cases.

FIGURE 2. (a) consists of input left/right images. (b) Is the final output generated by the proposed DAS which consists of proximity alerts along with corresponding navigation suggestions(**Danger! slow-down!! 2-wheeler in left** —>Better to be towards center Alert!! 2-wheeler in center Alert!! 2-wheeler in right). Refer to Figure 22, Figure 24, Table 4, Table 5 for more results on live traffic data.

adaptive masks are then post-processed for road spatial analysis based on their surrounding environments within the field of view range (θ^0_{FOV}) (13) to calculate proximity triggers and essential parameters required for framing robust navigation suggestions (refer to Tables 4 and 5 for sample results). Here in this proposed system, dense depth maps on input monocular 165^0 panoramic images are generated only during special cases (if all the l, c, r-FOVs in the host's front-view are blocked), The generated depth map is then processed to estimate essential parameters for framing precise navigation suggestions even in the above ambiguous states. Based on the above calculated object-proximity values/triggers, parameters & values of both front-rear views, final proximity alerts along with corresponding navigation suggestions are framed for the end user to ensure safer driving. The framed alerts, suggestions are shown to the user by displaying output on screens (used in this paper), through voice assistants, predefined vibrations, and FPGA probe signals [7], [6], etc.

The research in our proposed method contributes to the existing literature, and serves to the technology of driving assistance systems and autonomous systems by proposing novel AMMDAS. The presented research contribution has thoroughly surveyed many existing DAS trends and has identified several disadvantages and feature incapacities among them, to tackle those disadvantages we have brainstormed for various implementation features and have introduced several novel architectures & methods. Based on these features, architectures & methods, the proposed research work (''AMMDAS'') is framed and presented. The novelties & contributions of this proposed research are:

- We have proposed a novel adaptive mask generation & processing architecture for designing explicit proximity alerts and corresponding navigation suggestions.
- A novel pipeline is proposed which consumes homogeneous left, right stereo feed parallelly to generate an overall intuitive wide view for advanced driving analysis & assistance.
- We have proposed an extensive multi-threaded mutual analysis operation for 230^0 ultra-wide FOV analysis on combined left, center, right & rear field-of-views.
- We have introduced a new custom-built,[4] live traffic dataset (''Left, right synchronously stereo paired Indian on-road traffic dataset''), which will be open-sourced in the future. The proposed method requires an ensemble of a custom-built dataset, and multiple public datasets (MS_COCO, PASCAL-VOC, CU-Lane, KITTI) for training, validation, and testing of our proposed methods and architectures.
- We have proposed a novel encoder-decoder CNN (i.e D.S.E.D network) for advanced lane-modeling, along with corresponding custom-built loss functions.
- The proposed D.S.E.D network is an ensemble of 2 separate mini-ED & main-ED networks.
- The proposed mini-ED is used for a deep pre-processing mechanism, and the main-ED is used for external lane & FOV segmentation and analysis.

[4]Our custom built dataset ''Left,right synchronously stereo paired Indian on-road traffic dataset'' is liscenced and is subjected to no-objection during capturing under govt. permission and regulatory.

- An adaptive polygon modeling strategy for deep FOV analysis is introduced in D.F.O.V.M.
- An extra supportive depth-map analysis sub-module is proposed, to automatically handle uncertain ambiguous traffic conditions.
- This research has proved its effectiveness and superiority of our proposed AMMDAS, by evaluating its performance under multiple test wild case scenarios (section 5). Moreover, a series of qualitative and quantitative benchmarking analysis was performed on each internal sub-module and proposed method/architecture with other alternative state-of-the-art models.

The rest of the paper describes, illustrates, and evaluates the previously mentioned novel methods and research contributions. Section 3 describes our thorough research analysis performed on other existing alternative DAS systems. Section 4 details about our proposed research's methodology (in a stage/sub-module wise description). Section 5 demonstrates our proposed AMMDAS performance on live traffic conditions (including several wild case scenarios), moreover, section-5 also contains a detailed benchmarking analysis on each of our proposed methods & sub-modules. Section 6 includes the limitations faced by our proposed method and our future to overcome these limitations. Section 7 concludes our proposed AMMDAS's research work.

II. ABBREVIATIONS & ACRONYMS

ADAS: ''Advanced driving assistance systems''; AMM-DAS: ''Adaptive Mask Modeling Driving assistance system''; BOL: ''Bitwise-AND object lane''. 2D-Conv: ''2-Dimensional Convolution operation''; 2D-De-Conv: ''2-Dimensional De-Convolution''; D.F.O.V.M: ''Dynamic Field of View Modeling''; DSED: ''Dual Sequential encoder decoder''; DMAS: ''Driver management assistance system''; DNN: ''Deep neural networks''; DSSD: ''De-Convolutional SSD''; ED: ''Encoder-Decoder'' networks; FOV: ''Field of View''; FPN: ''Feature pyramid network''; FP: ''False positive''; FL: ''Feature Layer''; FN: ''False negative''; K: ''==l or r or c(represents a specific side of FOV)''; FM: ''Feature Map''; l: ''left''; c: ''center''; r: ''right''; l, c, r: ''left, center, right''; OMM: ''Object Mask Modeling''; PDF: ''Probability Distribution Function''; POI: ''person of interest''; ROI: ''region of interest''; R,G,B: ''Red, Green, Blue''; HSV: ''Hue, Saturation, Value''. RPN: ''Region Proposal Network''; SRV: ''spatial relief value''; SSD: ''Single Shot Detector''; SVD: ''Singular value decomposition''; TP: ''True Positive''; TN: ''True Negative''; YOLO: ''You Only Look Once''.

III. RELATED WORK

Current DAS systems use simple camera vision [4]–[7], [13], [15], [16], [18], based approaches or use multiple sensors like radars, LIDARs, high frequency GPS, proximity sensors etc [9]–[11]. There are methods & papers which utilized the above mentioned hardware on bottom bases and have shown some significant contribution towards DAS systems. Some methods take the input feed and process either in 2D [3]–[5], [7], [13], [16], 3D point cloud [6], [11], [14], [17], [18] format or in dense disparity depth maps each of the methods implemented show both advantage & spike in productivity along with disadvantages & increased production costs for their respective implementations. Moreover, some methods concentrate only on object flow [3], [4], [9], [13] and behavior analysis, some are biased towards a specific class [3], [4], [11] to identify proximity, and many methods analyze the correlation and occurrence between the objects present in FOV with respect to host source using 2D/3D image processing, disparity map processing techniques and some by implementing DNNs. Here H. G. Jung et.al [6], proposed a stereo vision-based obstacle detection and distance measurement method. By introducing traveling lane-based ROI establishment, peak detection by threshold-line, and edge feature correlation based verification, they were able to overcome the problems of the previous disparity histogram-based method and extend their applications to highway collision warning. Their paper had less FP, FN cases compared to their competitive papers. However, that performance gain required additional computation load. They were able to verify the improvement of disparity histogram-based obstacle detection by implementing a basic lane detection method; generally to create practical applications advanced lane detection methods are required (section 4.C). Furthermore, commercialization requires the FPGA implementation [6], [7], [18] of a core algorithm such as stereo matching, edge-feature correlation, and disparity histogram generation. This research work did not concentrate on identifying the type of Objects which are in proximity and their detection results showed sensitivity towards wild conditions (i.e improper lighting conditions (Fig. 24(d), (i)), shadows (Fig. 24(a), (j)), etc.). In Quin Long et.al. [14], Proposed a new methodology for identifying small objects in the roadway based on stereo vision. They used a multi path viterbi algorithm to obtain dense depth data of stereo images. From the generated depth information, the highway or roadway surface can be detected. Objects on the road can be mapped to the 3D space to determine their size and location. It can generate denser results, lower error rate, and faster speed compared to the native stereo matching algorithms which are widely used in intelligent vehicles' society. Besides, the developed research work did not consider broader and generalized FOV for spatial analysis and didn't frame any suggestions, this paper mainly has concentrated on detecting small objects which are in immediate proximity to the subject.

Work in Josip Cesic et.al. [9], proposed/addresses tracking & Detection of objects in motion within the context of ADAS [8]. They used a multi sensor setup consisting of radar and a stereo camera mounted on the vehicle's roof. They proposed to model the sensor's ambiguity in polar coordinates on Lie-Groups and apply the object's state filtering on Lie groups. To solve the multi target tracking issues, they

used a joint integrated probabilistic-data-association filter and presented necessary modifications to use on Lie groups. The proposed method was tested in live real-world data which was collected based on the above described multi sensor setup in urban traffic scenarios. The Limitation of this paper is that it mainly focuses on object detection and motion tracking, but does not concentrate on lane analysis and road spatial analysis to frame alerts and suggestions. The FPS achieved by this method is low to be in live. On the other hand S.Decker et.al. [10], Proposed a methodology based on fitting the model of a vehicle contour to both stereo depth image and radar readings by Concerning radar and stereo vision integration. First, the algorithm fits a contour from stereo depth information and finds the closest point to the contour with respect to the vision sensor. Second, it calculates the closest point of the radar's observation and fuses both radars, vision's nearest points. By translating the initially fitted contour to the fused the closest point, the resulting contour is located and obtained. Over and above this experiment has very restricted FOV for proximity alerts and the proposed algorithm is the best suitable for stationary vehicles with a predefined object flow and overall cost involved is more. As in Alberto Broggi et.al. [18], presented an Obstacle detection system and was successfully employed during VIAC (VisLab Intercontinental Autonomous Challenge), effectively negotiating a wide variety of scenarios. This approach was proven effective even in the presence of steep climbs and off-road areas. At the end of the expedition the algorithm has been further enhanced by the introduction of a high performance SGM implementation for the depth mapping stage. Calibration still remains a critical issue. To get good performance on obstacle-segmentation, the optical flow could be employed and an efficient implementation needs to be found in order to maintain processing time under control even on conventional hardware. This research [18] work includes a few problems: this method doesn't propose driving suggestions for safer navigation and does not consider surrounding the complete front, rear, left, right FOV and this method might fail to detect miniature & lower-scale objects in proximity and moreover it requires a higher configuration hardware setup. A.Howard et.al. [11] describes an integrated system for real-time detection and tracking of pedestrians [4] from a moving object. This paper used stereo vision as the primary sensor, and showed that this sensor has numerous practical advantages compared to monocular vision. These include the capacity to quickly identify ROI in an image, classify identified regions based on shape, and track detected pedestrians in 3D-real world coordinates. This system can also utilize monocular appearance based algorithms, using regions-of-interest with known scale to increase speed and reliability. Finally, stereo vision permits the creation of fast algorithms to detect independent motion in a scene. Compared with radar & LIDAR, the main weakness of stereo vision is its relatively poor range resolution. Moreover, this experiment is restricted to pedestrian analysis. Bihao Wang, et.al. [15], proposed an on-road object detection system and a strategy in obstacle extraction from U-disparity. Then, a modified particle filtering is suggested for multiple object tracking. Besides, multiple cues, such as obstacle's size verification and combination of redundant detections, are merged into the system to improve its accuracy. This experimental findings demonstrate that the proposed system is effective and reliable when applied to other multiple traffic video sequences from a public database. Nevertheless, the use of 2D coordinates has a certain limit for further localization and tracking of the obstacles and lane [12], FOV modeling isn't performed so this paper's method might fail and trigger false alarms to objects which aren't really in proximity when tested in curved roads.

The above discussed papers show sensitivity towards wild, noisy conditions, and do not consider a wide field of view ([3], [5]–[7], [9], [13]–[16]). Either in some papers, the input is a stereo feed where they merge the feed for their implemented algorithm to process (to create disparity or depth maps etc) [3], [5]–[7], [11], [15], [16], [18], by considering a single camera or stereo merged feed we can not capture the entire spatial surrounding in a single epoch for the algorithm to process. This may lead to feature incapacity when it comes to high sensitivity fault tolerant and reliable systems (see Fig. 1). Generally, let us consider input feed through ordinary cameras or existing DAS vision input approaches, where their FOV range is 80^0-90^0, this limited field of vision may lead to a false negative or false positive case for proximity alerting or during navigation suggestions when an object approaches from the extremities in the FOV (130^0-165^0& 0^0-15^0) Fig. 1(b). The proposed system tries to solve these issues, disadvantages & incapabilities (Fig. 25) which were identified with the existing DAS systems, by implementing several robust and reliable features & methods. The below mentioned features, and the modules/stages which will be discussed in future(section 4), are built based on the proposed research's contributions and novel architectures/methods(section 1).

Features Implemented in this proposed system (AMM-DAS):

- Precisely specify the type or class of objects which are in proximity.
- Considers Dynamic left-right-center & fixed rear FOVs.
- Correlative driving suggestions for safer navigation along with proximity alerts.
- The Proposed pipeline's architecture is modeled as a parallelized sub-modular processing architecture, to reduce the overall latency of the system without compromising in accuracy and robustness.
- FPS achieved by our method is suitable for on-live run.
- Wider 165^0 panoramic view as input, to overcome the FOV restrictions.
- Objects within the range are identified accurately to trigger proximity alerts precisely with low FP, FN rate even during wild external environments.
- A reliable adaptive Lane and FOV modeling system that generates appropriate FOV frontiers even in wild conditions.

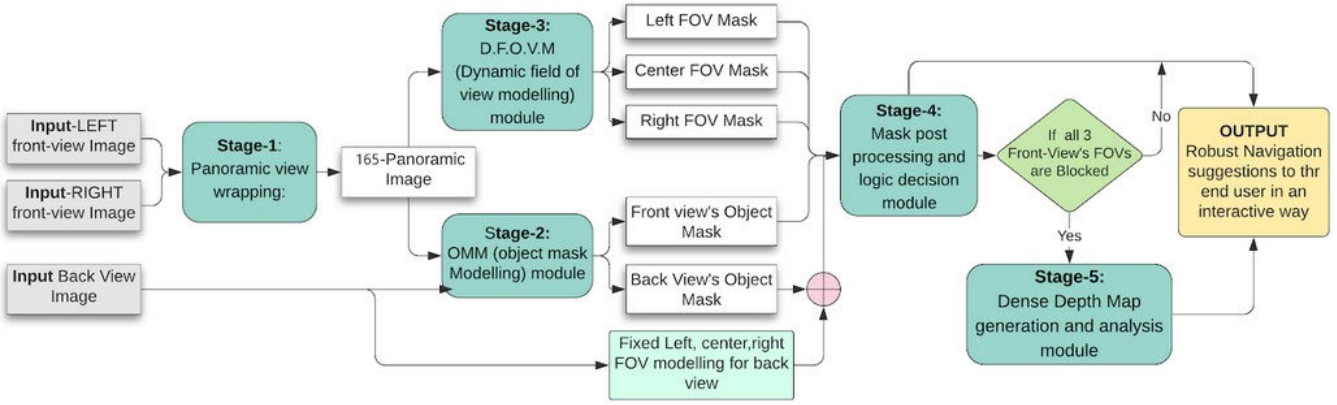

FIGURE 3. Overall system architecture of the proposed AMMDAS, which considers input from the front view's left, right stereo camera feed along with rear view's monocular camera feed. Now, the input is processed according to the above illustrated pipeline to dynamically output proximity alerts along with corresponding navigation suggestions.

IV. IMPLEMENTATION DETAILS ALONG WITH STAGE WISE DESCRIPTION

In this paper, the proposed DAS implements a unique pipeline of several sub modules that execute concurrently, according to Fig. 3. The proposed pipeline consists of several novel and best performing sub-modules/stages (in terms of accuracy and latency), to create an overall reliable and fault tolerant system. The proposed pipeline includes proprietary sub modules along with some corresponding research derivative sub modules. The derived sub-modules were brainstormed and benchmarked with other relevant methods (refer to section 5) so that the performance: latency-time trade off is robustly maintained. In this proposed DAS the entire pipeline consists of 5 stages, where each stage has its significant contribution in overcoming the challenges mentioned above. Pipeline's uniqueness lies in dynamically generating adaptive masks by processing and analyzing a wide input field of view (thus the proposed DAS is named as AMMDAS). Here Stage-1, 3, 4 are our proprietary, and stage-2, 5 are built from the inspiration of other research methods. The five stages which were included in the pipeline are:

- **Stage-1:** Panoramic view wrapping
- **Stage-2:** OMM module (object mask modeling)
- **Stage-3:** D.F.O.V.M module (Dynamic field of view modeling)
- **Stage-4:** Mask post-processing and logic decision module
- **Stage-5:** Dense depth generation and analysis module

The proposed system processes both front and rear views parallelly. In front view processing all 5 stages mentioned above are necessary and for rear view processing only stage-2, 4 are required, but final output alerts & suggestions are heuristically generated by considering both views. For the front view, temporal synchronous left and right input (left & right) images are fed to Stage-1 to generate a panoramic image (Fig. 4). The constructed panoramic image covers a whole sum of 165^0 wide field of view in a single step (Fig. 1), this stage helps to stitch a wide view of the surrounding environments so that bias towards a specific FOV is avoided, to make the overall correlation analysis in later stages perform better by involving more number of factors while framing navigation suggestions and proximity alert triggers, etc. After stage-1, the panoramic image is sent to Stage-2(OMM) and stage-3(D.F.O.V.M) simultaneously to generate object masks (Fig. 7) and left, center, right FOV masks (Fig. 12). By implementing OMM module, we can identify objects, POI in proximity with their respective class/type with high accuracy, i.e very low FP, FN cases, the sampling rate/FPS achieved by our implemented method is $\sim$2 which is optimal for live inferencing. D.F.O.V.M module generates adaptable FOV masks dynamically using a dual-sequential encoder-decoder network(DSED), the FOV masks are generated heuristically by considering surrounding objects and lanes present (left, right, center (Figs. 1 and 12)), so that the proximity alerts and navigation suggestions are framed on logically broad insight. Dimension/shape of a polygon(Δ^{le}) present in l, r-FOV masks is a function of lane-offset-value (Figs. 12 and 21); Δ^{le} in l, r-mask = f (l, r offset). The object mask and FOV masks from both front and rear view are merged to form Bitwise-and-object-lane(B.O.L) masks (Fig. 14), these post-processed masks are pixel & histogram analyzed, so that the Stage-4's logic module triggers proximity alerts and frames better logically correlated driving suggestions. An extra sub-moule i.e Stage-5 (Dense depth module) is added to the proposed pipeline when the system enters into a dilemma state (if all l, c, r-FOVs are blocked by vehicles). In this case, a dense depth map is generated from the panoramic image by a Monocular-depth estimation network (Fig. 15). A deep spatial analysis is performed on the generated Dense depth map to frame navigation suggestions. As stated above, in rear view analysis the input monocular camera feeds are sent directly to Stage-2: OMM module to generate Object Masks, now based on these generated object masks we perform Bitwise image anding with Fixed rear FOV masks to generate rear-view's BOL masks(B.O.L), The values extracted from this rear-view's B.O.L masks (i.e Object proximity triggers, spatial

relations) are concurrently sent with Front view generated values to Stage-4 to create insightful final proximity triggers and left, right, center, rear correlated navigation suggestions as discussed above. The proposed DAS operates on an overall 230^0 external FOV (combining front and rear view) to generate reliable navigation suggestions and reduce the sensitivity of error margin even during ambiguous situations. Now all the alerts and suggestions generated are sent to the end-user.

Sub modules/Stages (stated above) implemented in this paper require huge diverse, quality rich data for training and testing. The importance of high data requirement is to make the proposed DAS generate alerts and frame navigation suggestions in real time with minimal errors in multiple geographic locations with minimal initial calibration. Stage-1 requires our in-house proprietary dataset for camera calibration and metadata aggregation for effective wrapping of panoramic images. Stage-2 requires MS-Coco [19], Pascal-VOC [20], and our custom-built dataset for training and testing. Stage-3 requires Our custom-built Dataset, CU-Lane Dataset [21], KITTI [22] (modified these datasets to suit our requirements). Stage-5 requires KITTI [23], [24] dataset only for inferencing and tuning of generated-depth maps. Our custom-built dataset is made from videos recorded by left, right stereo paired cameras. Our dataset consists of: 1) rames extracted from left and right recordings; 2) Panoramic images stitched from left, right extracted frames using Stage-1; 3) plus raw traffic recordings from both cameras. The recorded videos are shot at 10 different geographic locations with a total of 200+ minutes (110 Kms). Extracted Left, right frames are of 1920*1080 Resolution with a total of 2,67,300+ images. The methods, DNNs, algorithms present in their respective stages in the pipeline architecture feed on the available data to process them and generate transitional outputs and parameters, which knits the entire pipeline to work in one to generate reliable navigation suggestions. Stages from 1-5 are discussed in detail below.

A. STAGE-1: PANORAMIC VIEW WRAPPING

$$G_{\sigma 1}(x,y) = \frac{1}{(2\pi\sigma_1^2)} e^{-\frac{x^2+y^2}{2\sigma_1^2}};$$
$$DoG \cong G_{\sigma 1} - G_{\sigma 2} \cong \frac{1}{(2\pi)}\left(\frac{1}{\sigma_1} e^{-\frac{x^2+y^2}{2\sigma_1^2}} - \frac{1}{\sigma_2} e^{-\frac{x^2+y^2}{2\sigma_2^2}}\right) \quad (1)$$

The 2 synchronous left and right frames are taken as input (Fig. 4(b), Fig. 16), and relevant features, keypoint patterns are refined and highlighted by applying the Difference of Gaussian(DoG) algorithm [25] ((1)). In DoG the original image is subtracted with a blurred version of it with a specific σ value, the blurred version is generated by convolving the original image using a Gaussian kernel. B(x, y)= w*O (x, y), here w is a Gaussian kernel which follows a Gaussian function property, $G\sigma_1$(x, y) is a Gaussian function with $\sigma 1$ standard deviation, by using this function the center pixel in the kernel gets higher weightage and the distant pixels gets negligible weightage with respect to the convolving pixel the kernel coverage depends upon σ value specified. g_1(x, y) - g_2(x, y) = Gσ_1 * f(x, y) - Gσ_2 * f(x, y) = (Gσ_1 - Gσ_2) * f(x, y) = DoG * f(x, y). In the DoG algorithm [25] we get a new feature enhanced image DoG(x, y) (1) by subtracting 2 Gaussian blurred images $g_{1,2}$ (x, y) (g_1(x, y) = Gσ_1(x, y)*f(x, y)) with different standard deviations(σ_1, σ_2).

As Feature enhanced left-right images are generated from DoG, these images are now used to find left and right key points and detect Local invariant descriptors using the Harris Corner Detection(HCD) [26] algorithm (Fig. 16(c), (d)). In this proposed paper, HCD [26] is used rather than SIFT [27] because, the raw & processed input images are of same dimension (and resembles to same scale) and these both images captured are under similar physical conditions, so a generalized lite-Feature-Descriptor algorithm is picked, to make the panoramic image stitching operation execute on-live with optimal time & space complexity. So, based on the above factors [27] HCD algorithm is used. HCD algorithm [26] finds corresponding contours, edges in the input images. In HCD, corners of contours, shapes are identified in the image where there is a significant change in intensity function (2) in all possible directions.

$$f(\Delta x, \Delta y) = \sum_{(x_k, y_k)\in W} (I(x_k, y_k) - I(x_k + \Delta x, y_k + \Delta y))^2 \quad (2)$$

f(Δx, Δy) is the intensity change value between the specified neighborhood (N-8 [28] which is represented by W), if f(Δx, Δy) is near to 0 then that region/surface is of same intensity and if f(Δx, Δy) is of a larger value then a corner at that point is identified. The intensity differences along x, y-axis are calculated by using second order Laplacian difference [28], I_x is the intensity difference along x-axis and I_y is intensity difference along y-axis. $I_x = DoG^x\sigma_{1,2} * I$; $I_y = DoG^y\sigma_{1,2} * I$; Using these I_x, I_y values $I_{x2,y2,xy} \in I_{x2} = I_x.I_x$, $I_{y2} = I_y.I_y$, $I_{xy} = I_x.I_y$ are calculated. The above stated I values which were calculated across neighborhoods are also used to calculate (3) M (structure tensor) matrix.

$$M = \sum_{(x,y)\in W} \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix} = \begin{bmatrix} \sum_{(x,y)\in W} I_x^2 & \sum_{(x,y)\in W} I_xI_y \\ \sum_{(x,y)\in W} I_xI_y & \sum_{(x,y)\in W} I_y^2 \end{bmatrix} \quad (3)$$

Here, W is the neighborhood considered. Using M matrix, Corner Response value which is R=Det(M)-K(Trace(M))^2 is calculated. This calculated R value is solely dependent on the Eigenvalues(λ1, λ2) of The Structure-Tensor(M), The R value is large if a corner is identified in the Neighborhood W, R value is -ve in an edge and small in plain & flat regions. To finalize the key Descriptors in the image for image mapping, the R value is calculated for every pixel and now thresholding with a specific limit is applied so that all flat regions, edges are eliminated. After thresholding Non-Maximal-Suppression [29] is applied so that only thick-relevant, confident descriptors remain. Using

the Key Descriptors identified in both Left, Right images (Fig. 16(c), (d)) now map these both left, right input image's descriptors and combine them with the help of parameters and homography matrix.

Algorithm 1 RANSAC (Random Sample Consensus) Algorithm for H Homography Matrix Calculator

1. RANSAC(a,b- Key point pairs matched using K-d-Tree+HCD):
   Input: all the N {a,b} K-d Tree matched key point pairs.
   Output: H homography matrix parameters.
2. **For** i = 0 to N **do**
3. Select 'n'-random {a,b}-Key-point pairs
4. Compute homography H parameters(using S.V.D & Direct Linear transform) based on $\{a_i b_i\}$ keypoint pairs
5. Generate plausible inliers with a conditional restraint of
   if $S.S.D(p_i^{|}, H.p_i) < R$ **then**
   Estimate the p_i, $p_i^{|}$; plausible inliers
   else
   continue
   end if
   where R is a specified range Constant through the process, and H is homography matrix, p_i, $p_i^{|}$ are points on their respective images
6. Generate a registry of inliers with respect to i^{th}-iteration.
7. **end for**
8. Now, Re-compute the least-squared error H estimate($\|p_i{*}H\text{-}p_i^{|}\|$) of all N terations's inliers generated above, Here least squared error H-estimate acts as a performance evaluation error metric.
9. **return** H-homography matrix
10. **end**

The homography matrix and parameters are generated using the RANSAC algorithm [30]. The Feature Key-points between the images are matched using Arc-Similarity/ Euclidean Distance($d(a, b) = (\Sigma[a_i - b_i]^{0.5})$) using the distance similarities, the robustness of the matched Key-points aren't reliable as the dimension of the key-points are very large, so Neighborhood-matching strategies like (N.N.D.R), K-D Tree [31] are considered. N.N.D.R is a distance ratio (d_1,d_2) between the nearest neighbors of respective points ($= (d_1/d_2)$), if N.N.D.R value is small then the 2 points are said to be similar. N.N.D.R is a computationally overloaded operation as it bruteforce the distance ratio calculation between all possible points; the optimal method for Key-point matching is to use the K-D tree algorithm [31]. Now using the optimized [29] matched set of Key-points between the L, R input images, the left and right images are projected to a Projective (combine input-images) plane using a Homography-matrix(H) (Fig. 4(a)), the parameters in the H-matrix are calculated using the Ransac Algorithm.

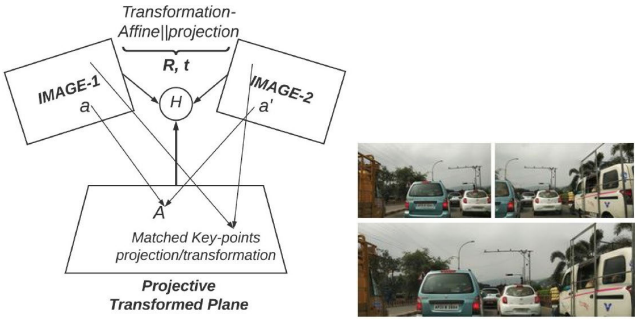

FIGURE 4. (a) a, a' are the input stereo images which are being projected into A(P.T) plane using R, t transformations. (b) input left, right stereo images are being mapped into a final 165^0 panoramic image.

RANSAC is an N-iterative random method which generates robust fault tolerable parameters so that they best fit the input mathematical-model.

Using the H matrix parameters and Keypoints, the images are now affine/projective-transformed [28] onto a perspective plane. $a = C.[Rt].A$, $a^{|} = C^{|}.[Rt].A^{|}$, C is a real-plane projection parameter, a, $a^{|}$are key-points in images which are mapped to A in P.T(projective transform) plane. So by using keypoints, H($H = C^{|}.R^{|}.C^{-1}.R^{-1}$) is calculated and determined using SVD. Using P.T(projective transform) plane and H parameters the panoramic and calibration parameters are determined. Using P.T plane, H, panoramic and calibration parameters as inputs, a multi-resolution image registration [32] is performed. Image-tuning is done to the registered PT plane image using above calibration parameters to minimize the distortion artifacts, exposure variances and chromatic aberrations. Final wrapping of the overall view is done to compost [32], [28] the image into a mono RAW-wide view. To generate the final 165^0 panoramic view (Fig. 16(e), (f)) poisson-blending [33] + interpolation-fit [34] is applied to the mono RAW-wide view.

B. STAGE-2: OMM(OBJECT MASK MODELING MODULE)

The panoramic output from stage-1 is passed as input in this stage. This module is inspired from faster-RCNN with the Feature Pyramid Network(FPN)] [35] as backbone. In overview the input panoramic image is passed to this model and Bounding Box(BB) values along with respective class id [36] are generated, now using these generated class-id & BB values dynamic Object masks are generated. For feature extraction backbone, FPN [35] is chosen because small objects at different resolution scales which are in the input field of view should be detected (Fig. 18(u), (w), Fig. 7(e)), no object must be skipped in detection so that proximity alerts and suggestions are correlated better (refer to Fig. 17 for performance comparison). FPN network [35] is a multi scale, pyramid architecture where it extracts features & patterns of every scale. FPN outputs multi resolution Feature Maps, where at each specific resolution, objects of a specific scale are identified with higher confidence (Fig. 7(e)). FRCNN+FPN [35]–[37] is designed using

multi-pyramidal-scale feature extractor & RPN(region proposal network) with accuracy and speed as concern. Using a general FRCNN single scale Feature-maps(FM) [36] are generated for Region-Proposal-Network(RPN) so by considering this single scale feature-map, objects (especially small and very big) aren't identified and FP cases arises as there wouldn't be any consensus of that specific object at a different view. To overcome this FRCNN+FPN is implemented to generate Multi-Scale-feature-maps. A FPN [35] consists of both Bottom-up and Top-down path proceeding or flow (Fig. 6). Generally the Bottom-up flow is a Convolution Neural Network(CNN) which is pre-trained on IMAGENET [38], this CNN is used to extract Feature maps from the input image given. In general CNNs as we go deep into a network, Shallow Features at first becomes more semantically concentrated later at deeper layers, and the overall resolution of the image decreases as we apply convolution [39] & Max-pooling [40] multiple times at multiple layers (by going deep/bottom-up resolution decreases and Semantic concentration increases). In object detection both Feature refinement with semantically condensed patterns and feature-resolution are important for good predictions and results. When shallow layers are considered, resolution is good but feature refinement & semantic values are compromised, and therefore no. of FP cases rise [41]. If deep layers are considered, the feature resolutions are low but features are semantically rich and therefore low resolution objects are neglected, and no of FN cases rise [41]. FPN implemented in this paper combines both bottom-up and top-down pyramidically [35] so that an even composition of semantic-value and resolutions at different scales are generated. In a Top-Down flow higher resolution feature layers are reconstructed and interpolated from Dense semantic-value feature-maps of bottom-up flow. Here as the input images are up & down sampled during bottom-up and top-down flows the spatio-temporal integrity between the feature maps and original image is lost. To overcome this problem Lateral-connections between these two flows are established (Fig. 6). These lateral connections help to maintain the Spatial correlation between the Corresponding Feature Maps (in bottom-up flow) to the respective reconstructed layers (in top-down flow). In Bottom-up flow, any Deep-CNN [37] pre-trained on imagenet [38] can be used. Here in this paper ResNet-101 network [37] is used.

This ResNet-101 [37], [35] consists of 5 clustered Residual-Blocks(C_i), where the output from each clustered residual-block is a set of intermediate-feature maps labeled as C_i(i $\in$ 1-5). Each residual block consists of many 2D-Conv + ReLU + B.N+ Max-Pooling operation layers. 2D-Conv layers [39] are used to extract high-concentration features from the input features. ReLU [42] is used to maintain integrity and to introduce Non-Linearity among the Features, Batch-Normalization(B.N [43]) is used as the FPN has skip/residual connections between top-down and bottom-up layers therefore feature normalization must be performed after adding a skip/residual connection [35] for normalizing the features and preventing the FPN from overfitting. Max-Pooling [40] is used for dimensionality reduction. For every C_i Residual block the dimension of feature decreases to its $\frac{1}{2}$ and respective-convolution stride increases. C_i generated in Bottom-up is laterally connected to the M_i in top-down path. In top down 2D-1*1-Conv(FM(x, y, d)) = FM^1(x, y, d^1) = M5 is applied, to reduce the depth of C5 to 256-d, the corresponding result after this 1*1-conv operation yields M5. M5 is the first layer of output-scale feature-maps (P5). As we proceed deep in top-down flow, lateral connections from bottom-up flow and previous M_i are concatenated. Here ($i \in 4, 3, 2, 1$).

$$P_i = 2D - 3 * 3 - Conv(Upsamp\,(M_{i-1}) + 2D - 1 * 1 - Conv(C_i)) \quad (4)$$

M_{i-1} is up-sampled (*Upsamp*) by 2x factor for each P_i ((4)) to match the dimension while concatenating. A final 2D-3*3-Conv is applied to a concatenated Feature layer to nullify aliasing effects which occur due to up-sampling. Generally for FPN in an object detection module only P_5, P_3, P_2, P_4 are considered because P_1 has high spatial dimension and lower feature, semantic values (same as features at shallow layers in DNN), due to this FP/false predictions may occur [41]. The above generated multi scale(pyramid) feature maps($P_{5,4,3,2}$) are now fed to Region Proposal Network(RPN) [36] to find out ROIs of objects present in an image. RPN consists of Conv layers + anchor-proposals (Fig. 5), These anchor proposals define ROI of objects, patterns. ROIs act like attention mechanisms that tell the FRCNN [36] to look for patterns, objects. In RPN a sliding window is hovered throughout the P_i Pyramid-scale Feature maps. Each sliding window is sent to 2 FCN [44] (Softmax-classifier and ROI regressor). These 2-FCN [44] layers generate multiple possible ROIs using anchors [36] (similar to SSD, YOLO etc.) Anchors help to predict multiple regions in a single window so that the resulting ROI orientations can be of any format & scale. Here in this paper multi-scale-anchor boxes are implemented so that regions, objects of multiple scales can be identified in a window, Generally a Softmax-classifier and ROI Regressor outputs 4K-BB parameters and 2K-class scores (class $\in$ 1||0; object||not-object), where K is max number of proposals(anchors), K∞Scale&Aspect-Ratio [36]. Scale & Aspect-ratios used in this paper are mentioned below. ROIs in this FRCNN-FPN [35], [36] can be of multiple scales, sizes and orientations within a single instance. RPNs can be optimized by implementing an efficient loss (5) function ($L(\{P_i\},\{T_i\})$ [36]). While training; p_i represents the probability score whether the respective i^{th} ROI is an Object or not. T_i represents the boundaries(BB) of the i^{th}ROI. N_{cls}($\in$256), N_{reg}($\in\sim$ 2400) are normalization parameters. λ($\in$10) is a balancing constant. L_{cls} is a log loss [*-(klog(p) + (1-klog(1-p))* $\in$ 1||0-classification] and L_{reg} ($==L_\delta$) is a robust loss (5), Where 'a' is the difference between the IOU of t_i, t_i^*, δ is a

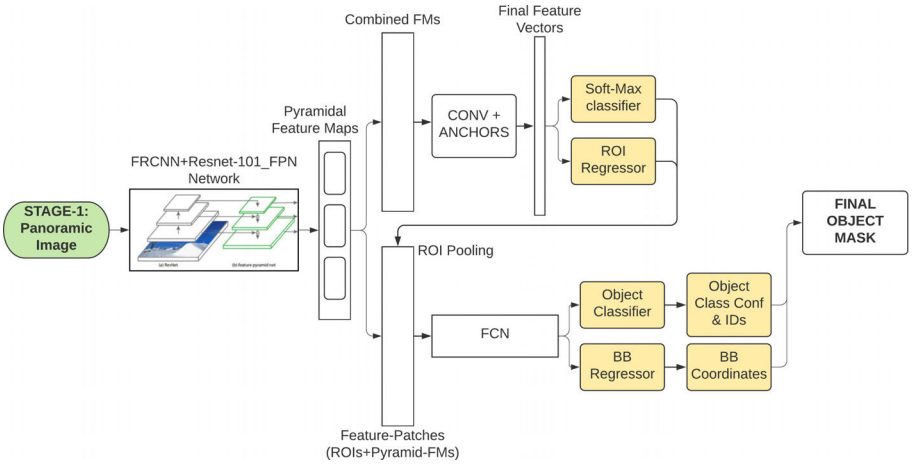

FIGURE 5. Architecture overview(steps/methods & execution) of OMM module with FRCNN+Resnet-101_FPN [35], [36] network as an object detector.

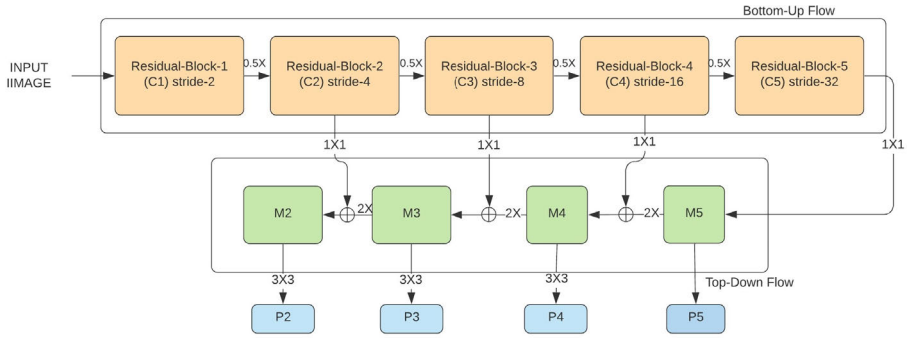

FIGURE 6. FPN architecture [35] illustrates top-down, bottom-up architectures, and corresponding lateral connections between them. Where an input image is fed to the FPN network to output multi-resolution feature maps(Pi) respectively.

tolerable range of difference in (5) (further detailed in [36]).

$$L\left(\{p_i\},\{t_i\}\right) = \frac{1}{N_{cls}}\sum_i L_{cls}\left(p_i, p_i^*\right) + \lambda\left(\frac{1}{N_{reg}}\sum_i p_i^* L_{reg}\left(t_i, t_i^*\right)\right)$$

$$L_\delta(a) = \begin{cases} \frac{1}{2}a^2, & for\ |a| \le \delta \\ \delta(|a| - \frac{1}{2}\delta), & otherwise \end{cases} \quad (5)$$

Each of the ROIs are generated with respect to a specfic pyramid-scale feature-map. Based on the size of an i^{th} ROI a specific pyramid scale level(K); $K = [K_0\{ \in 4\} + Log_2((w.h)^{0.5}/224)]$ is assigned. [35] Here, K represents scale offset value used in P_k-ROI pair tagging, w & h are respective width and height of a ROI (i.e if a ROI with specific width, height value gets $k = 2$, then that ROI is tagged along with P_2 (4)). After mapping i^{th} ROI (k==j) with P_j, these $\{i^{th}$-ROI and $P_j\}$ pairs are sent to the ROI-Pooling layer where respective Feature-patches are generated. These Feature-patches from ROI-pooling are sent to Fully connected layers [36], [44] to get a specific object-class present in that respective feature-patch with Bounding Box(BB) around it. The Bounding Box is refined via BB-regressor so that the IOU with respect to ground truth is maintained optimal during predictions. During training each feature-patch is sent to FC layers and BB-regressor units, to calculate optimal Class-id and BB-coordinates. BB-regression is performed by minimizing the cost-function (between $[t_x, t_y, t_w, t_h]$ & $[t^*x, t^*y, t^*w, t^*h]$) [36], x, y are center coordinates of a specific Bounding box and w, h are their respective width, height. In (6), $t_{x,y,w,h}$ are predicted Bounding box parameters and $t^*_{x,y,w,h}$ are ground truth values. So from the above formulation and discussion when an image is given as input respective object classes along with BBs are generated, but the generated Bounding Boxes are redundant, so we apply Non-Maximal Suppression [29] to extract final Bounding-Boxes for object-classes(o$\in$ O) to make the object detention network's predictions more reliable. So when an input image (Fig. 7) is given to the FRCNN+Resnet-101_FPN network [35]–[37] for prediction, respective pyramid-scale feature-maps are generated by the FPN, and similarly relevant ROIs are proposed with the help of RPN. Now, a pooling operation is applied on the above generated Pyramid-scale-feature-maps and ROIs, to generate optimal feature patches, and these feature patches

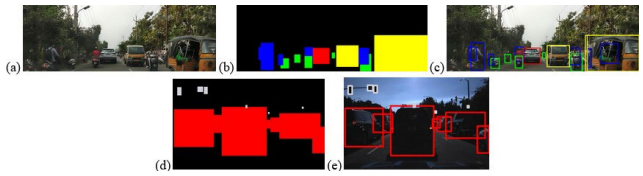

FIGURE 7. (a) is the input panoramic image given to the OMM module, and (b), (d) are the stage-2's final output object masks, and (c), (e) are intermediate object detection results generated by our object detector, FRCNN + Resnet-101_FPN [35]–[37].

are sent to FC layer [44] and BB-regressors((6)) to output final Bounding-box coordinates (Fig. 7(c), (e)) along with corresponding object-classes present in it.

Based on the Object-classes (o∈ O with respect to Table 2) and bounding boxes generated on a specific input, an Object-Mask(Fig. 7(b), (d)) with predefined color coding (mentioned in Table 2) is generated.

$$t_x = \frac{x - x_a}{w_a}, \quad t_y = \frac{y - y_a}{h_a}, \; t_w = \log\left(\frac{w}{w_a}\right),$$
$$t_h = \log\left(\frac{h}{h_a}\right), \quad t_x^* = \frac{x^* - x_a}{w_a}, \; t_y^* = \frac{y^* - y_a}{h_a},$$
$$t_w^* = \log\left(\frac{w^*}{w_a}\right), \quad t_h^* = \log\left(\frac{h^*}{h_a}\right) \tag{6}$$

In an Object Mask the area (with respect to i^{th} BB coordinates) under a specific i^{th}-object is filled with the respective o^{th} object-class's color, and rest of the background-region, space left is colored in Black (Fig. 7(b), (d), Fig. 18). FRCNN+Resnet-101_FPN network [35]–[37] is trained on i7-9th gen CPU coupled with 1-GTX-1070 GPU. In this paper, MS-Coco [19] + Pascal-VOC [20] pre-trained FPN & RPN networks are taken, and these pre-trained [19], [20] models are stitched to a FRCNN to form FRCNN+Resnet-101_FPN network, and this complete network is trained iteratively(∼15K steps) on our custom dataset (with 7-object-classes(O)) to detect traffic-specific-objects. During training, we have maintained these hyper-parameters accordingly (lr=0.0001, momentum=0.9, weight-decay=0.0005 and Sliding window-scales = [128, 256, 512] with [1:2, 1:1, 2:1] aspect ratios for anchor ROI-proposals).

C. STAGE-3: D.F.O.V.M (DYNAMIC FIELD OF VIEW MODELING MODULE)

In this stage(D.F.O.V.M) adaptive field of view masks are generated dynamically by analyzing external surroundings (lanes, objects etc) of the host. Extent of field of view covered by the D.F.O.V.M is given by $\Theta_{FOV}{}^0$(explained later in this stage (13)). Generated FOV masks are dependent on a Lane-Segmented embedding image. This Segmented-embedding image (Fig. 12) is heuristically generated by our Dual-sequential Encoder-Decoder network. Dynamic field of view modeling module(D.F.O.V.M) takes input from stage-1, to process and output FOV masks. The D.F.O.V.M and stage-2(OMM) will always run in parallel. The main elements of D.F.O.V.M module are Dual-sequential Encoder-decoder network (an encoder-decoder CNN [45]), Lane-point clustering algorithm [46], adaptive l-r offset prediction operation and Dynamic l, c, r-FOV modeling [28] module. Each element mentioned above executes in a sequential order by consuming respective outputs of previous elements as their input (Fig. 8). The overall flow followed in D.F.O.V.M is, a 165^0 panoramic image is given as input to the Dual-Sequential-ED network (Fig. 8) to output a lane-segmented Embedding image which has similar dimensions and spatial relations to that of the input image. This output lane-segmented embedding image is passed to the Lane-point clustering algorithm [46] for analysis & post-processing, so that it generates essential parameters, which are then passed to l-r offset prediction method to heuristically calculate adaptive l-r offset values, so that these offset values can be consumed by the FOV modeling module to output final Left, center, right FOV masks (Fig. 12). These FOV masks are the final outputs generated by the entire Stage-3(D.F.O.V.M) module. Now let's discuss each element present in the D.F.O.V.M module in detail which were discussed above. Dual sequential Encoder-Decoder network has 2 components present in it, which are mini-ED (Fig. 10) and main-ED (Fig. 11). mini-ED is used for thorough pre-processing and fine-tuning of raw input images (Fig. 9). When an input image is given to the mini-ED network, it tries to eliminate (blurs or fades-out) all unnecessary patterns & regions in the image, and highlights only essential parts & patterns in the image (Fig. 9(d), (e), (f)), which helps in boosting the overall performance & accuracy of the DSED network. In naive terms mini-ED acts as an attention model where it highlights contexts, feature-spaces which belongs to both lanes & objects-on roads (or roads themselves; Fig. 9 (d), (e), (f)). This pre-processed input is then fed to the main-encoder-decoder (main-ED) to output the final lane-segmented embedding image (Fig. 12), where segmentation is done with respect to a fixed color coding in which the dataset (refer toTable 2) is prepared for training and testing (Fig. 9(g), (h), (i)).

$$BN\,(x_i) = \hat{x}_i^{(k)} = \frac{x_i^{(k)} - (\frac{1}{m}\sum_{i=1}^{m} x_i^{(k)})}{(\frac{1}{m}\sum_{i=1}^{m}(x_i^{(k)} - (\frac{1}{m}\sum_{i=1}^{m} x_i^{(k)}))^2 + \varepsilon)} \tag{7}$$

Main operations performed in DSED network are 2D-Conv [39], 2D-Deconv [47], Batch-Normalization [43], ReLU [42], Max-pooling [40], Feature-Concatenation and softmax function. Batch-Normalization(BN) [43] is used for normalization and to deal with co-variance shifts, so that integrity and stability are maintained during forward and backward passes of trainig iterations. This is done by normalizing the feature layer by adjusting the weight/activation units with respect to an input batch according to (7). k ∈ [1, d] and i ∈ [1, m], d is the dimension of the feature space/vector,

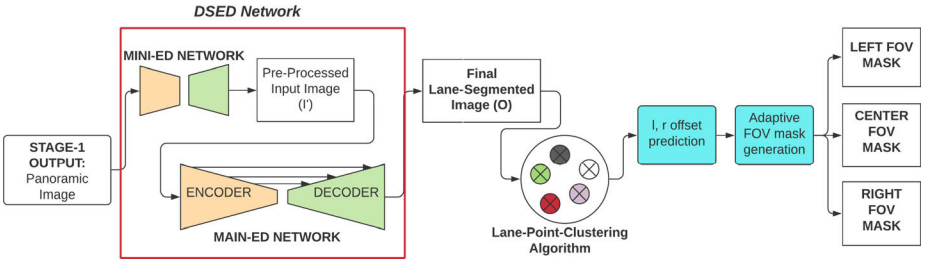

FIGURE 8. Architecture overview(steps/methods & execution) of Dynamic field of view modelling module (D.F.O.V.M).

and i is the size of the batch in that iteration/epoch. $y_i^{(k)} = \gamma^{(k)}x_i^{(k)} + \beta^{(k)}$ so $x_i^{(K)}$ is replaced by $y_i^{(K)}$; $(y^{(K)} = BN_{\gamma,\beta}(x^{(K)}))$. 2D Convolution(conv [39]) or De-convolution (de-conv [47]) ($H^1 = W*H = \sum_{i\in N} W(i).H(i)$) operations are used for feature extraction process(H^1) by convolving a kernel/filter(W) on an input feature layer/maps(H). When multiple input units ($\in$ N; neighborhood in W) have to be mapped to a single output unit(down-sampling) then Conv [39] operation is performed, and when a single input unit has to be mapped to multiple output units(up-sampling) then De-conv [47] is applied. Here Convs extract/produce higher semantic valued & concentrated feature maps. De-convs are used in Decoder parts [45] to extract higher level spatial & categorical features and also to increase the resolution of the respective input feature maps. ReLU [42] ($max[0, x]$) introduces non-linearity among the output feature maps, to make weights/activations more suitable and stable (tackles vanishing gradient) for optimization [48] during back propagation. Max-pooling [40] ($max\{W_i\}$ [i $\in$ n*m; n, m = dimensions of neighborhood]) is a sample Discretization method, which is used to extract sharp and smooth features, and also to reduce dimensions of the input features, so that the chances of overfitting are minimized. It picks the max value present in the neighborhood of non overlapping regions (stride = n, for n*n-max-pooling operation). Feature concatenation operation is used when 2 feature maps/layers should be merged together into one, the dimension of features which are to merged should be same, to maintain integrity among dimensions cropping and applying 1*1 convs are done.

Softmax is used to normalize the distribution of input feature vectors (logits of feature activations) into a probability distribution((8) pd(x)), N is the depth of input feature maps/feature layers, a_K is the activation unit value in the K^{th} feature layer at x^{th} pixel location, pd(x) is the probability distribution of x^{th} pixel location among n-depth feature layers. Based on the above discussed base-functions, the entire DSED network is built.

$$pd\,(x) = \frac{exp^{a_k(x)}}{\sum_{k1=1}^{K} exp^{a_{k1}(x)}}; \quad k \in [0, N] \tag{8}$$

Overall Mini-ED network has 8-feature layers. mL_i represents i^{th} feature layer. **I', O'** are the input panoramic image and output embedding image respectively with H*W*3 dimensions. mL-1 consists of 96 feature maps (depth), and is generated by applying 2D-5*5-conv(I') [39] followed by BN [43], ReLU [42] & 2*2 Max-pooling operations; and $mL_{2,3}$ are of 256, 1024 in depth and are generated by applying 2D-5*5-conv($mL_{1,2}$) + BN + ReLU + 2*2Max-pooling operations. $mL_{1,2,3}$ belongs to the encoder network [45] and $mL_{4,5,6}$ belongs to the decoder network [45] (Fig. 10). In decoder network, the highly semantically condensed feature maps of mL_3 are deconvolved to increase & extrapolate its spatial resolution & condensed feature maps for reconstructing an output image, with similar dimensions to that of the input image. Logical inter-correlation & co-occurrence between the current & previously extracted features/patterns is maintained intuitively throughout the decoder network. As we go deep into the decoder network,resolution of a respective mL_i feature-map increases correspondingly by reducing its features-depth. mL_4, mL_5 & mL_6 have 256, 128 and 64 feature maps respectively which are generated by performing $mL_i = 2D\text{-}5 * 5\text{-}De\text{-}conv(mL_{i-1}) + BN + ReLU$ *operations*. We apply 2D-1*1-Conv operation (to condense 64 FMs to 3 RGB channels) to mL_6, to convolve through the feature layers to output **O'** feature layer (Fig. 10). This **O'** is passed to main-ED network for processing and to generate the final lane-segmented-embedding image. Conv and De-Conv [47] are convolved on feature maps with auto-padding and 1-stride, and the kernel/filter dimensions are 7*7 throughout the entire mini-ED network so that smooth & cogent feature maps are generated. We follow a 2-phase training mechanism for the

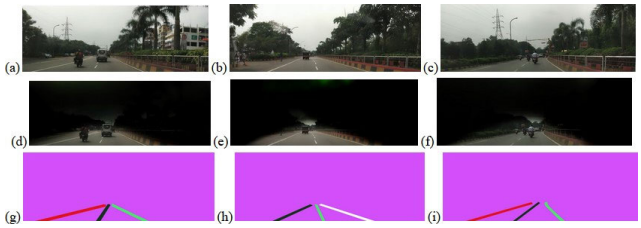

FIGURE 9. These are the sample ground truth labeled images present in the training data of the DSED(mini-ED+main-ED) network. (a), (b), (c) are the sample raw input images present in dataset. (d), (e), (f) are the sample ground truth images for training the mini-ED network, and (g), (h), (i) are the sample ground-truth lane-segmented images for training the main-ED network.

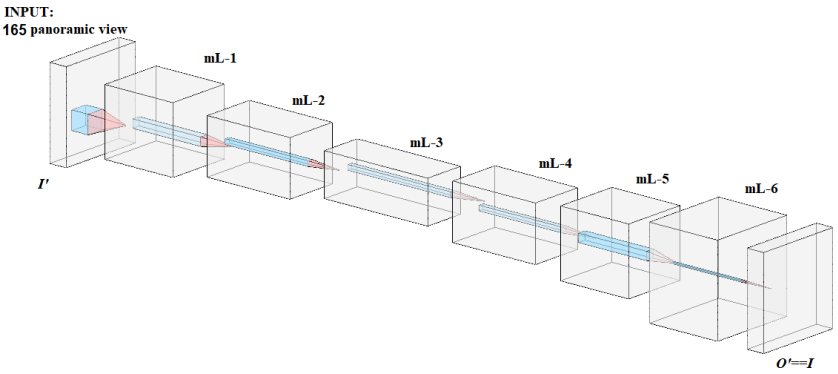

FIGURE 10. Mini-ED architecture, this proposed encoder-decoder network takes HXW panoramic input(I^I), and outputs pre-processed intermediate embedding image(O^I) with same HXW dimensions, here mL-i represents i^{th} feature-layer of the mini-ED network, where mL-1 to mL-3 belong to the encoder network, and mL-4 to mL-6 belong to the decoder network.

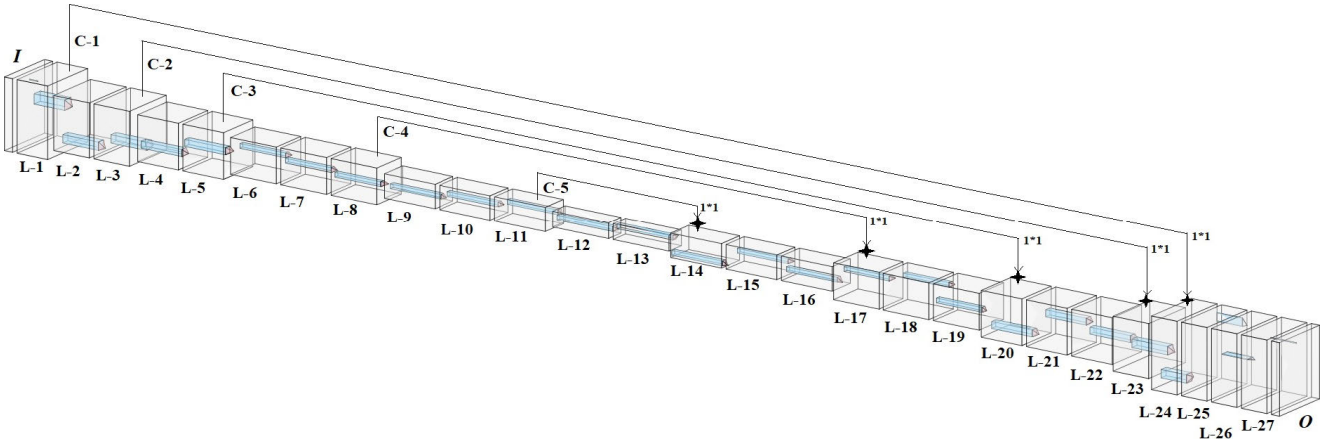

FIGURE 11. Main-ED architecture, this proposed encoder-decoder network takes HXW pre-processed image from mini-ED(I), and outputs final lane segmented embedding (O) with same HXW dimensions, here L-i represents i^{th} feature-layer of the main-ED, and C-i represents i^{th} lateral connection from the encoder network(L-1 to L-13) to the decoder network(L-14 to L-27).

entire DSED network, where mini-ED is initially(phase-1) trained separately, and then mini-ED+main-ED networks are trained together(phase-2) with custom & predefined loss functions. While training the mini-ED network in Phase-1, we calculate(L_{sqe}) per pixel-wise Squared error loss $\{(L_{sqe} = \sum_i^{N*M} ||OP_i - GT_i|)^2$; N, M $\in$ dim(G.T, OP)$\}$ between the output(OP) and Ground Truth(GT) images, and this calculated penalizing value is optimized using SGD optimizer [48]. mini-ED network is trained on 15,000 images (labelled according to Fig. 9(d), (e), (f) format); these 15K images are fragments of our custom-built dataset. The pre-processed image O' (output of mini-ED) is now passed to the main-ED network. Main-ED is a deep convolutional encoder-decoder network [45] with lateral connections between the encoder and decoder networks. This network has 29 layers with $L_{12,13}$ feature-layers serving as a bottleneck(latent feature maps) between encoder and decoder networks. L_i represents i^{th} feature-layer. **I** is input pre-processed image(output of mini-ED) & **O** is the final lane-segmented image, these both I, O images share same dimensions (H*W*3). To explain main-ED we split the entire network into encoder-network and Decoder-network (Fig. 11) parts. In the encoder-network(L_{1-13}), depths of Feature-Layers gradually increase and simultaneously their respective feature resolutions also decrease, and the extracted features become more concentrated and semantically/feature rich. Encoder network has 13 Conv encoding layers with multiple filter sizes, where each 2D-conv operation [39] is followed by Batch-normalization [43] and 2*2 max-pooling [40] operations, For some 2D-conv layers a lateral connection(C_i) is given to the decoder network. L_1 feature layer is obtained by performing 2D-1*1-conv operation to map the input image of 3 RGB layers to 64 feature-maps (Fig. 11). L_1-FL operation is essential in main-ED network because, the features in the input images will be distributed accordingly into sparse feature layers with more depth, so that feature extraction per layer when convolved further will be more pattern specific and distributed. Both of the $L_{2,3}$ FLs have 128 Feature Maps(FMs) and are generated by applying $2D\text{-}11*11\text{-}Conv(L_1) + BN + ReLU + 2*2\text{-}Max\text{-}pooling$ and $2D\text{-}7*7\text{-}Conv(L_2) + BN + ReLU$ operations respectively. Generally to get any Feature map in the encoding network

we apply the same equation(9) but with different filters/kernel dimensions.

$$L_i = f_{N,M}\,(L_{i-1}) = 2D - N * M - Conv\,(L_{i-1}) + BN + ReLU; \{N, M \in \dim\,(kernel)\ \&\ i \in [1-13]\} \quad (9)$$

$L_{4,5}, L_{6,7,8}, L_{9,10,11}, L_{12,13}$ have 256, 512, 1024, 2048 feature Maps(FMs) respectively. Each layer(L_i) is generated by applying the above mentioned sequence of operations((9)) on L_{i-1} with respective kernel dimensions at specific layers. $L_4 = f_{7,7}\ (L_3)$, $L_{5,6} = f_{5,5}\ (L_{4,3})$ and $L_{7to13} = f_{3,3}\ (L_{6\ to\ 12})$ respectively. Now a 2*2-max-pooling operation with 2-stride [40] is applied only on $L_{1,3,5,8,11}$ layers to get $L_{2,4,6,9,12}$, so that $L_{2,4,6,9,12}$ FLs have smooth and distinctive features, and also to minimize the chances of overfitting ($dim(L_{2,4,6,9,12}) = dim(L_{1,3,5,10,11})/2$) while training the entire D.S.E.D network. Based on the above discussed operations at each layer, a complete encoder network is constructed. In the Decoder-network, the bottleneck feature layers($L_{12,13}$) along with lateral connections($C_{i,}$) from the encoder network are processed & mapeed together to generate O^t (Final lane-segmented embedding image). The Decoder network consists of 14 convolutional reconstruction layers; from L_{14} to L_{27} the resolution of each feature layer gradually increase and their respective depth(i.e no of FMs) gradually decrease, as we apply De-conv operation at each i^{th} layer. There are 2 sequences of operations((10) g^1, (11) g^2) applied here in this decoder network.

$$L_i = g^1_{N,M}\left(L_{i-1}, C_j\right) = 2D - 1 * 1 - Conv(C_j \oplus 2D - N * M - De - Conv(L_{i-1})) + BN + ReLU \quad (10)$$

$$L_i = g^2_{N,M}\,(L_{i-1}) = 2D - N * M - Conv\,(L_{i-1}) + BN + ReLU \quad (11)$$

$\oplus$ is a Feature concatenation operation where Lith layer from the encoder network is appended to Deconvoluted features layers of Li-1th decoder network. In g1(), Cj (j ∈ 1, 2, 3, 4, 5) represents a lateral connection from L_i^{th} (i ∈ 1, 3, 5, 8, 11) encoder network's layer. A 2D-Deconv operation is applied on L_i (i ∈ 13, 16, 19, 22, 23) and the resulting feature layers are concatenated ($\oplus$) with C_j, after concatenation a 2D 1*1-conv is applied to condense the depth of FLs to half. The condensed features are batch normalized [43], and then ReLU activation [42] is applied (10) to output L_i (i∈ 14, 17, 20, 23, 24). Unlike in Segnet [49] or De-convnet [50], [47] we do not store max-pooling indices to perform Up-Sampling+De-conv operations, because by reconstructing an output image only from bottleneck features (generated by an encoder network and also by sampling spatial indices(i.e switch keys)), we miss minor, dull & broken lane lines in a 165^0-wide panoramic input image.

To support the proposed use-case and to perform robust during on-live-traffic inference, we need a network that considers features of all resolutions and semantic concentrations. Therefore, during decoding the bottleneck feature maps(i.e decoder network of main-ED) we parallely add features with similar resolutions thrown from the encoder network via C_j, similar to U-net [51]and FPN [35]. Here in this proposed main-ED network, we don't crop feature layers which are to be concatenated, instead we merge the features as it is, and perform 1*1conv mapping to condense the depth of feature-maps so that the border lanes & necessary patterns aren't affected, and also co-occurrences and spatial relations between features are maintained homo-geneously throughout the network. L_{14} has 1024 FMs and is generated by $g^1 3,3(L_{13}, C_5)$, and $L_{15,16}$ also have 1024 FMs and are generated by $g^2 3,3(L_{14,})$, $g^2 3,3(L_{15})$ respectively. Similarly, $L_{17,18,19} = g^1 3,3(L_{16}, C_4)$, $g^2 3,3(L_{17})$, $g^2 3,3(L_{18})$ respectively with 512 FMs; and $L_{20,21,22} = g^1 5,5(L_{19}, C_3)$, $g^2 5,5(L_{20})$, $g^2 5,5(L_{21})$ with 256 FLs. $L_{23}(g^1 7,7(L_{22}, C_2))$ and $L_{24}(g^1 7,7(L_{23}, C_1))$ have 128 and 64 FMs respectively. After $L_{23,24}$, multiple 2D-Conv operations [39] are applied to L_{24} to generate $L_{25,26,27}$, because at $L_{23,24}$ low level features are appended from the encoder network, these lateral connections have higher spatial resolutions and low feature/semantic values & density. So multiple feature extraction operations have to be performed so that these concatenated shallow features become more refined & semantically concentrated with relevant features extracted. All $L_{25,26,27}$ Feature layers have 64 feature Maps ($g^2 7,7(L_{24,25,26})$), and a 2D-1*1-Conv is applied to L_{27} so that all 64 activation-values/units in i^{th}(i∈ *[0-n*m; n, m=dim(L-26)]*) pixel location at respective layers are mapped (pd(x)) and fed to multi-label softmax classifier (K=7 {no of classes=7}) to classify features vectors accordingly (lane classes are mentioned in Table 2), to finally generate an output lane segmented image(**O**) with H*W*3 (input165^0-panoramic image) dimensions. Main-ED is combined with phase-1 pre-trained mini-ED to form a complete DSED network. For training this complete DSED (phase-2 of training), we have used our custom-built dataset along with CU-Lane [21] dataset. KITTI road lane dataset [22] was used only for validation and testing. Our custom built dataset consists of 2,67,300+ live traffic images. Out of these 2,67,300+ images 95,000 images (fragmented part of our data, specially used for this module) are considered for training and 5000 images for validation. Ground truth labeling & segmentation of the training and testing data is done according to Table 2 color format. DSED network takes panoramic images with resolution of ∼(2400X900*3) as input and generates a lane segmentation map with similar dimensions. Using the above prepared data (*100K(custom data) + 10K(CU-Lane) + 0.5K (KITTI road lane dataset)*) we train our network with lr =0.0001, momentum = 0.99. Per pixel Cross-entropy ($L_{CE} = -\sum_{j=0}^{N*M}\sum_{i=0}^{K} y_{i,j}.\log(p_{i,j})$; {$K \in$ *no of classes* == *7, N, M = dim(L27)*}) with respect to generated Semantic map based on softmax-classifier prediction & ground truth is calculated, so that this loss value(L_{Total}) is minimized by using Adam optimizer [48], [49]. To converge this network to a optimal loss value(1.059 in our case) it took 11,500 epochs.

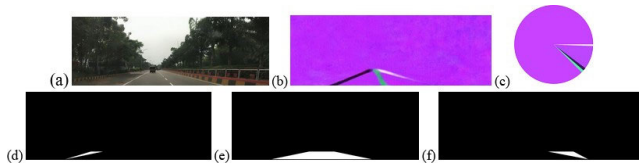

FIGURE 12. (a) Input panoramic image from stage-1, (b) is the final-Lane segmented embedding image generated by the DSED network. (a) input panoramic image consists of right, immediate right and left lanes, so the output embedding image(b) contains semantic segmentations (color coding mentioned in Table 2) based on the respective lanes identified by the DSED network. (c) is a pie representation of the clusters formed by the lane-point clustering algorithm [46], when a lane segmented embedding image is given as an input. (d), (e), (f) are the left, center and right FOV masks, and these l, c, r-FOV masks are D.F.O.V.M's final output. White contours present in l, c, r FOV masks are the polygons which are modeled based on calculated l, r-offset values.

A batch size of 32 is maintained during the entire training phase, and data augmentation is performed for every batch during epoch iterations. This network(D.S.E.D) is trained multiple times (with minor tweaks in hyper parameters and α, β values; best performing values were mentioned above) and the best performer on validation data is considered. During every epoch, the entire D.S.E.D network is penalized based on L_{Total} loss value.

$$L_{Total} = \alpha.L_{abs} + \beta.L_{CE} \tag{12}$$

where $\{\alpha, \beta = [1.5, 1]$ *for* 500^{th} *epoch and* $\alpha, \beta = [0.1, 1]$ *for rest of the epochs*$\}$, L_{Total} (12) is optimized by Adam optimizer [48] iteratively. This custom cost function calculates absolute error between the ground-truth l_{gt}, r_{gt}-offset values and predicted l_{pred} , r_{pred} offset values, $L_{abs} = ||(l_{gt} - l_{pred}) + (r_{gt} - r_{pred})||$ {l ∈ left-offset value, r ∈ right-offset value}. These l, r- offset values are predicted using parameters generated by a post-processing operation applied (which is done by Lane-point clustering algorithm) on the final lane-segmented image (Fig. 12). The Final lane-embedding image generated by the DSED network trained on CULane [21], custom-dataset is now sent to the Lane-point clustering algorithm(by K-means [46]). Here we cluster the (R, G, B) pixel colors of the final lane-segmented embedding image (each pixel's RGB color, value refers to predicted lane-class), into 5 clusters (Fig. 14(c)) to determine the quality, type of lane lines present and identified. The centroids of 5-clusters are R, G, B-lane segmentation colors (white, green, olive green, red, purple shade) which are used in G.T Dataset preparation for training and validation processes (refer to Table 2). The set of 5 centroids along with their respectively clustered pixels (Fig. 14(c)) are sent to the next element in D.F.O.V.M to predict l, r-offset values (Tables 4 and 5). Using these parameters we predict adaptive l, r offset values. The l, r offset values are functions of a lane-class present in an i-th cluster, and number of pixels which belong to that specific cluster. Initially l, r offset values are assigned to 0, and then we iteratively update the l, r values accordingly for every cluster present in the resulting set (output of Lane-point clustering algorithm [46]). If an i^{th} cluster ∈ white(right-lane, [255, 255, 255]) then *l-offset* = *l-offset* + *1.5* + $P(px_{white})$ and r-offset remains unchanged. Similarly, a cluster ∈ red(left-lane, [255, 0, 0]) then *r-offset* = *r-offset* + *1.5* + $P(px_{red})$ and l-offset remains unchanged. Now if i^{th} cluster ∈ green (immediate right-lane, [0, 255, 0]) then *r-offset* = *r-offset* + *0.75* + $P(px_{green})$ and *l-offset* = *l-offset* + *0.25*, and if a cluster ∈ olive-green(immediate left-lane, [20, 50, 25]) then *l-offset* = *l-offset* + *0.75*+$P(px_{olive-green})$ and *r-offset* = *r-offset* + *0.25*. Here $P(px_i) = ((no.\ of pixels\ in\ i^{th}\ cluster / total\ no\ of pixels))$. The role of P(px) in FOV mask modeling is, if an object is present in a respective i^{th} lane area then the no-of pixels contributing to that i^{th} cluster class will be less, resulting to small $P(px_i)$ value, and if no objects are identified in an i^{th} lane area then the no-of pixels in that specific cluster class will be more so $P(px_i)$ value will be large. Based on this analogy if a respective lane has no object in it, then we span more FOV towards that lane (large l/r-offset value because respective P(px) is large) to search for objects more comprehensively, so that alerts and suggestions will be framed on a larger FOV for safety & robustness.

Similarly, if a lane has an object present, then we consider less FOV towards that lane, as we would like to avoid the host to get in close proximity towards that object, and give more emphasis for other alternative FOVs to frame a safer suggestion for navigation. Using these l, r offset values we now construct dynamic FOV masks for left, center, right views. The center FOV mask always consists of a constant polygon, and left, right FOV masks have adaptive polygons (an obtuse angled triangle) whose areas/shapes are dependent on l, r-offset values. The base of Δ^{le} in Left FOV mask = Base_span*l-offset, and base of Δ^{le} in Right FOV mask = Base_span*r-offset. Upon various experimentations, we have fixed Base_span value to 150pixels and Height_span value to 450pixels for optimal performance. The position of polygons in Left, center and right FOV masks (Fig. 12(d), (e), (f)) are placed according to the position of lanes in the Lane-segmented embedding image. In this paper we have coined a FOV range parameter named ($\Theta_{FOV}{}^0$), whose value corresponds to the overall FOV/surroundings range covered by the generated adaptive left, center, right FOV masks. $\Theta_{FOV}{}^0$ (13) is the overall instantaneous FOV range(in^0) covered by the AMMDAS (Figs. 22 and 24) to adapt & analyze its external factors so that robust results and suggestions can be generated. Fig. 12(d), (f) images represents left and right FOV masks respectively, and Fig. 12(e) represents center FOV mask. White region/contour present inside each FOV mask, indicate adaptive polygons which are drawn & modeled [28] based on base and height values calculated above (using l, r offset values). L, r masks are spanned according to host's external surroundings. In Fig. 12(f) the polygon present inside the right FOV mask is large compared to the polygon inside left-FOV mask because, evidently in Fig. 12(a) there is more space & leniency towards the right FOV side (i.e there are immediate-right, right-lanes present). So we consider more FOV towards right side to

comprehensively search for objects and simultaneously analyze right-FOV's external surroundings to heuristically generate navigation alerts and suggestions. Moreover, left side is restricted because only immediate-left lane is present. So based on the above considerations we span less FOV towards the left and give more emphasis towards center & right FOVs, for logically better and robust results. More results and experimentation of D.F.O.V.M module is described in Figs. 22 and 24, Tables 4 and 5 of section 5.

$$\theta^{0}_{FOV} = (\tan^{-1}\left(\frac{150 \times l - offset}{450}\right) + \tan^{-1}\left(\frac{150 \times r\_offset}{450}\right) + 55)^{0} \quad (13)$$

Algorithm 2 LANE-POINT CLUSTERING Algorithm

1. LANE-POINT CLUSTERING) $\{O_{i,j}\}$; i ∈ n, j ∈ m; n,m = dim(O)):
   Input: Final lane segmented image O_{nxm}
   Output: $\{PX_{ij}\}$ j represents any of the cluster center(i.e nearest lane class GT RGB-color), i ∈ [0, N]
2. Group N pixels(px) according to R,G,B intensity values where N = n ∗ m
3. Initialize number of clusters k = 5
4. **While** True **do**
5. **if** C_ith centroid == $C_{(i-1)}$th centroid **then**
6. break
7. **end while**
8. **else**
9. Assign 1 cluster center per class(*in*5) at random order. So that we get 5-centers
10. **for** i = 0 to N **do**
11. Assign PX_i; to the closest cluster center(C) using L2-nomi/ Euclidean distance L_2-norm $\sum_{j=1}^{k}\sum_{i=1}^{k}(RGB\,(px_i^j - c_j))^{0.5}$, $RGB(px\text{-}C) = R(px)\text{-}R(C))^2 + (B(px)\text{-}B(C))^2 + (G(px)\text{-}G(C))^2$
12. **end for**
13. **end if**
14. Calculate centroid for N pixels in K clusters.
15. **return** $\{px_{ij}$: i ∈ N, j ∈ Kth cluster, px is RGB pixel in O image$\}$.
16. **end**

D. STAGE-4: MASK POST-PROCESSING AND LOGIC DECISION MODULE

Object masks from Stage-2 (Fig. 14(b)) along with left, center, right FOV masks (Fig. 14 (f), (g), (h)) from Stage-3 of both front and rear view are taken as input in this stage(Fig. 13) for post-processing [28], and to generate proximity alerts along with corresponding navigation suggestions. These framed suggestions are logically-correlated by passing the input for post-processing, and then to an internal logic-module for analyzing the interdependent spatial attributes & correlating the patterns & parameters between left, center, right FOVs. The post-processed masks are generated by performing perpixel-image anding [28] between respective kth-FOV mask and object mask. K-Post_processed mask = I^K_{N*M*3}, where I^K is a Kth pre-processed FOV image with N-width, M-height and (r, g, b) planes; $I^K[i, j] = k\text{-}FOV\text{-}mask[i, j]$ & $Object\text{-}mask[i, j]$ $i \in [0, N]$, $j \in [0, M]$ and '&' refers to bitwise AND [28]. K refers to the left, right and center field of views, as stage-4 takes l, c, r FOV masks from Stage-3 as input to post process them [28]. I^K_{N*M*3} (K Post_processed masks) are referred to as Bitwise-and-object-lane (B.O.L) masks in this paper (Fig. 14 (i), (j), (k)). These l, c, r-BOL masks are color histograms analyzed [28] to determine required parameters for the internal logic-module to frame proximity alerts and navigation suggestions. The parameters which are determined immediately after post-processing and histogram analyzing a specific [28] K-BOL mask are grouped together, as a list of several sets of pixels$\{PX^o\}$, which are calculated using (14). Where each set contains a group of pixels belonging to a specific Oth RGB color code; and the overall list consists of "i" number of pixel sets where i represents objects identified in that respective K-BOL mask (Fig. 14(c), (d), (e)).

$$PX^O_i = \frac{|K - BOL|^2_{i,o}}{Wh^K}; \quad i \in O^{th}\,[R, G, B]\,color\&Wh^K = |K - FOV|^I_{(I \geq 230,230,230)} \quad (14)$$

In Equation (14), $|\quad|^1$ is the cardinality of a set of pixels in K-FOV which belong to white color I $(K\text{-}FOV[i, j]) = (r, g, b) >= [230, 230, 230]$, (I is intensity of i, jth pixel in K-FOV mask). $|\quad|^2$ represents the cardinality of a specific ith pixel-set in the entire K-BOL's parameter-list, i∈ Oth object predefined (R, G, B) color code used in Stage-2, O has 7 different object classes each with a unique r, g, b-color scheme (refer to Table 2). Based on the above discussion a total of 2 sets are calculated, one set for front view and another for rear view, now these both sets are passed as inputs to the internal logic module. Where each set consists of 3 vectors/pixel-sets (∈ left, center, right), and each vector consists of 7-parameter values ($[PX^O], O \in [1 - 7]$) which belongs to a respective K-BOL mask. Initially in the Logic module pruning of input sets is done to yield better outputs. Pruning operation consists of trimming unnecessary values/parameters, here in this case all zero-valued parameters are pruned and only non-zero parameters in a vector are considered ($V_{1,2,3} \in [PX^O], PX^O! = 0$) for further processing (analysis of pruned vectors in sets). Here the vectors in a set are processed to yield final compounded l, c, r parameters. These compounded l, c, r parameters of each set(front and rear) are analyzed mutually to generate final proximity triggers along with corresponding navigation suggestions. Here the final output is generated only after logically correlating all of its input FOVs. Each parameter(PX^O in (14)) in the pruned vectors of

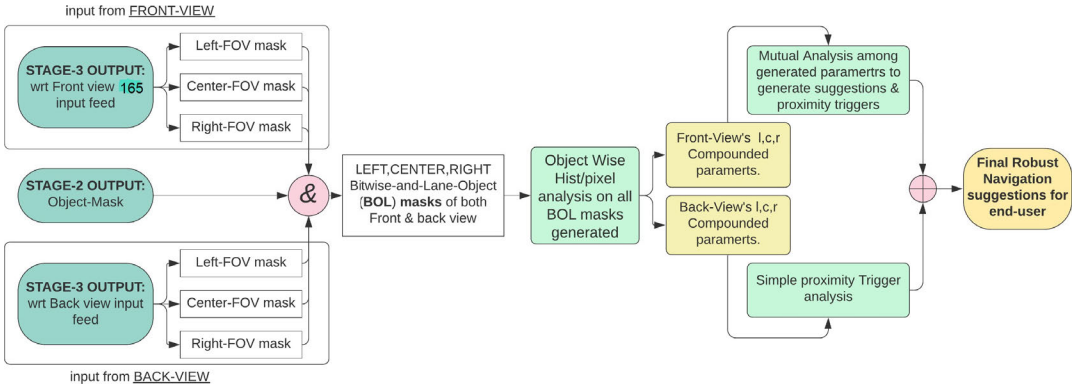

FIGURE 13. Architecture overview(steps/methods & execution) of Mask post-processing and logic decision module.

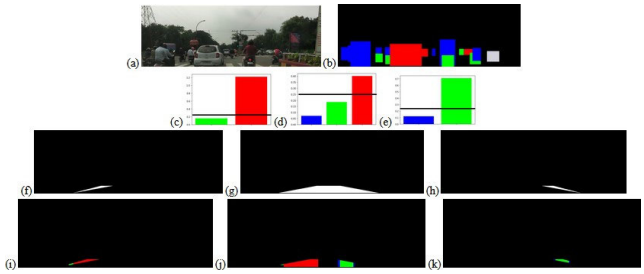

FIGURE 14. (a) is stage-1 output, and (b) is stage-2's final output when (a) is given as an input. (c), (d), (e) are pixel and hist distribution of l, c, r-BOL masks. Based on these distributions pruning, mutual analysis and thresholding [28] operations are performed to calculate compounded l, c, r-parameters, and (f), (g), (h) are l, c, r-FOV masks generated by D.F.O.V.M when (a) is given as input, and (i),(j),(k) are stage-4's internally generated left, center, right-BOL masks when (b), (f), (g), (h) are fed as input.

each front & rear view sets are categorized into three classes by thresholding them to a specific value(Th==0.25). The categorized classes are "no-alert", "alert" and "danger". If an ith $PX^O >= Th$, then ithPX is classified into "danger" or to "alert" if value is $0.05 <= PX^O < Th$ and if $PX^O < 0.05$ then it is categorized into "no-alert".

After categorization, a copy of pruned sets containing vectors of both views is made, so that the copied version consists only of each parameter's category (no-alert(-1), alert(0), danger(1)) at their respective places. Now these 2 sets (one with parameter values and another one with category classes) of a single view are sent for processing to yield respective compounded l, c, r parameters. Generally a compounded l, c, r parameter is a dictionary with K-FOV (left, center, right) as keys, where each key's(K) value consists of a subset of {float-value$_K$ & a sub-list}. Here, float-value$_K$ = (max{PX^j}; j O), O represents the distribution of total object classes(7 classes) as mentioned in Table 2, and j is a specific object class identified in each K-BOL mask (Fig. 14(c), (d), (e)). Sub-list consists of object-classes identified in its respective K-BOL along with their respective parameter's category(-1, 0, 1). The generated compounded l, c, r parameters are now analyzed to generate a final suggestion. In our paper only Front view's l, c, r-compounded parameters are Mutual analyzed and the rear view's compounded parameters are just checked only for proximity triggers & alerts. Logically, end users only need to know objects' proximity alerts from the rear view so that they can steer and move accordingly while moving forwards. So based on this logic front view's FOVs are very crucial, and they must be analyzed deliberately, therefore mutual analysis is performed only on the front view and rear view is just used for triggering proximity alerts. In mutual analysis of compounded l, c, r parameters, each parameter belonging to a K(left or right or center) is simultaneously analyzed with other FOVs($\in$ K) of the same set. We will now detail steps involved while framing suggestions and proximity alerts to the end user; Consider front view's compounded l, c, r parameters in which, if a K-key's value has all -1("no alerts") in its sub-list then we just simple append "**Nothing in K FOV**" to the Final suggestion. On condition that if a left or right FOV key is skipped in the front-view's compounded parameters, we then append "**Please keep towards K^1-FOV**" to the final suggestion (K^1 = left if right FOV is skipped and vice versa, K^1 $\notin$ K). If a parameter(PX^j) belonging to a sub-list of K's value is of 0("alert") category then we append "**Alert!! {jthclass object} in K FOV**". Now if a parameter(PX^j) in K-sub-list belongs to 1("Danger") class then we mutually analyze which Q-FOVs (K$\notin$ Q) have less object density comparatively, by checking the mini of Q-keys float-values (Q_X = *min(Q_1-max{PX^j}, Q_2-max{PX^{j1}}); j, j1 O*), Q$_{1,2}$ belongs to either of l, c, r FOVs other than current K FOV(K Q$_{1,2}$ = {left, center, right}). Based on the above mutual analysis we append "**Danger! slow-down {jthclass object} in K FOV; Better to be towards Q$_X$ FOV**" to the Final suggestion (Figs. 22 and 24). For rear view, simple proximity alerts with respect to K FOV are appended to Final navigation suggestion, i.e if a sub-list present in K key's value of rear-view's l, c, r-compounded parameters has a PX^j which belong to 1-"Danger" category then we append "**Danger!! {jthclass object} behind in K-FOV**" to the final suggestion. Generally the final navigation suggestion is heuristically framed by appending processed results of every K-Key's values present in respective l, c, r compounded parameters of both views (see Tables 4 and 5 for final outputs),

The proposed method was able to achieve robustness [41] because, each and every parameters in k-key's value is collectively analyzed and processed intutively. Final navigation suggestions generated from this Stage-4 consists of, object's proximity triggers (which were identified in the host's total FOV range($\Theta_{FOV}{}^{0}$)) along with navigation suggestions for the end user to help in avoiding fatal damages & accidents during driving. Sometimes Stage-4 enters into a dilemma state, when parameters of all K-keys values present in the front-view's compounded parameters consist of 1("Danger") category (Fig. 23), so performing mutual analysis operation in these cases is an illogical decision as every FOV is blocked with an object's proximity or due to high traffic flow. So, to tackle the above case (all FOVs blocked) we introduce an extra submodule(Stage-5), where the input 165^{0} panoramic image is directly consumed in order generate a dense depth map [11], [14], [17]; and robust navigation suggestions are framed by processing these dense depth maps.

E. STAGE-5: DENSE DEPTH MAP GENERATION AND ANALYSIS MODULE

This stage is optional, and is entered only when all the left, center, right FOVs trigger proximity alert above a specified threshold, otherwise the end-user gets final navigation suggestions framed by the Stage-4. In this stage a custom post-processed [28], [52] dense depth map (Fig. 15) is generated using a network inspired from this paper [53], which generates a dense depth map(Monodepth2) by self supervising a monocular vision. In Monodepth2 [53] they overcame the limitation of requiring per-pixel ground truth data by implementing a self supervised model, and they have proposed 3 architectures (depth, pose & full-res-multi-scale) with custom loss functions which are auto-masking loss and min-re-projection loss. These custom loss functions are used to handle and optimize occlusions and camera-motion assumptions. Monodepth reduces image artifacts by using a high resolution-multi scale sampling mechanism the overall method described is trained on a trino image (I_0, I_1, I_{-1}) set of the KITTI dataset [23], [24], and has achieved a benchmarking performance compared to other depth generation networks (Table 1). In monodepth2 [53] the current frame of the trino set is sent to the depth network to create a depth map. Depth network, CNNs, shared-encoders are pretrained on Imagenet (they have achieved SOTA performance even with pretrained models). Previous & next frame(I_1, I_{-1}) are sent to CNNs and pose network (pose network consists of encoded which encodes I_1, I_{-1} separately). The output from pose network($T_{t\ t'}$) and the depth map generated from depth network(D_t) are sent to appearance loss (min-re-projection & auto-masking losses) & full-res-multi scale sampler units to generate a final dense depth map (Fig. 15(b)) with minimal occlusions and artifacts [28]. For this paper we chose Monodepth2 because they have SOTA depth generation methodology and high performance(F_β) [41] with minimal loss values, when benchmarked with other depth-estimation methods on multiple datasets (KITTI-Odometry [22]–[24] depth prediction, Make3D etc), Monodepth2 [53] has RMSE loss of 4.701 which outperformed other depth-estimation methods [54]–[57] with a margin of 1.0^{+}on Make- 3D [58], Table 1 benchmarking analysis.

TABLE 1. Make-3D [58] benchmarking(Abs Rel, Sq Rel, RMSE, $\log_{10}$) on different depth map estimation methods.

Method	Abs Rel	Sq Rel	RMSE	$\log_{10}$	F_β -score
Monodepth	0.544	10.94	11.760	0.193	0.828
Zhou	0.383	5.321	10.470	0.478	0.912
Karsch	0.428	5.079	8.39	0.149	0.897
DDVO	0.387	4.720	8.090	0.204	0.941
Monodepth2	**0.322**	**3.589**	**7.417**	**0.163**	**0.983**

We have custom evaluated Monodepth2 [53] ($F_\beta = 0.98$) along with other methods on our custom dataset (Table 1) by calculating F_β -score($\beta = \mathbf{2}$) [41] using (12). Our custom evaluation dataset consists of 100 images taken under live traffic scenarios, where the host is subjected only to higher object-flow scenarios with all l, c, r-FOVs blocked. A common post-processing algorithm [28] is applied on all the depth maps generated by different methods, the output of the post-processing algorithm is either left or right. So based on the classification results of different methods, a confusion matrix is built to calculate F_β -score [41]. Here left, right classes are the input trigger parameters for framing logical suggestions which tells the end user to follow a specific direction/side on road (or) in traffic for a safer navigation (Table 5). We have used a pre-trained model of Monodepth2(M) [53] (trained on KITTI [23], [24]) and the avg FPS achieved by using Monodeth2 is ∼10.

$$F_\beta = \frac{\left(1+\beta^2\right).TP}{\left(1+\beta^2\right).TP + \beta^2.FN + FP} \tag{15}$$

The Depth map (Fig. 15) generated by Monodepth2 is now post processed for binary classification. In the post processing algorithm the depth map is cropped (Fig. 15(a)) so that spatial analysis is constrained only to relevant regions, so that logical suggestions can be generated. FOV restriction/cropping is an important operation because we avoid unnecessary regions (i.e skies, very far objects (traffic lights, trees, city buildings, etc)) in the generated depth-map. Moreover, relevant-FOV extraction helps the over-all algorithm (Stage-5) to run faster as analysis would be performed on a restricted part of image rather than the whole image. In Fig. 15(a), the red marked region is our interested cropped FOV in the entire input panoramic image. After relevant FOV extraction we perform depth-map texture conversion & fine-tuning operations. In texture conversion methods we convert the RGB-depth map to HSV format [28] depth map (Fig. 15(c)); and fine-tuning method involves a sequence of image dilation and erosion operations ($I^1 = (((I \Theta B)\ B)\ B)\ \Theta$), *where* Θ*=erosion,* $=$*Dilation operation*) B is a 3*3 matrix with all ones, except B[1,1](=2). These sequences of erosion and closing operations [52] helps in strengthening borders & patterns of objects, and also to cover up and link small void-contours & edge gaps created during texture conversion

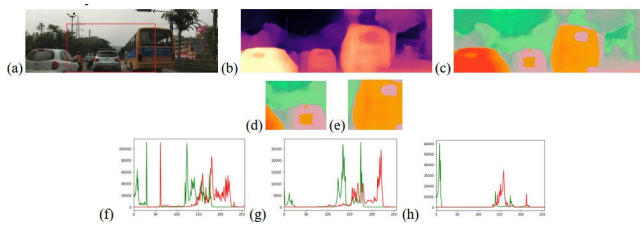

FIGURE 15. (a) in an input panoramic image, (b) is a dense depth map generated by the Monodepth2 network [53], and (c) is the HSV formatted depth map, (d) is the left FTC image and (e) is the right FTC image [28]. (f), (g), (h) are histogram distributions of (c), (d) respectively. Based on the analysis performed on (f), (g), (h) histogram distributions Red, Green, Yellow$_{PDF}$ values are calculated, and using these PDF values left, right SRV values are determined.

operation. Image smoothing operation on I^1 is performed using a convolution Gaussian kernel (with $\sigma = 3$) [28] to output a final fine-tuned cropped image(FTC). Now the above generated FTC image is divided into 2 equal halves, left FTC image (Fig. 15(d)) & right FTC image(Fig. 15(e)). Generally in I^1, Red spanned [$i \in$ *(255, 0, 0) -(220, 30, 30)*] pixels resembles nearer pixels, Green spanned [$j \in$ *(0, 255, 0) -(80, 230, 80)*] pixels resembles farther pixels and Yellow spanned [$k \in$ *(255, 255, 0) -(235, 160, 160)*] pixels resembled middle range pixels. We perform histogram analysis (Fig. 15(g), (h)) on left & right FTC images to calculate respective R, G, B probability distribution functions, i.e Red_{PDF} ($\Sigma_i PDF(i)$), $Green_{PDF}$ ($\Sigma_j PDF(j)$), $Yellow_{PDF}$ ($\Sigma_k PDF(k)$). These PDF values are used to determine respective Spatial relief values using (16) for both left & right FTC images. Based on the above discussed methods and calculated parameters we suggest the end-users either to "**be towards left**" if *Left-SRV>Right-SRV*, or to "**be towards right**" if *Left-SRV<=Right-SRV* . Generally only navigation suggestions are generated from stage-5 and the suggestions usually start with "*****All FOVs are blocked!! Suggesting based on a calculated depth map***".

$$l, r - SRV = l, r - Green_{PDF} + 0.25 \times l, r - Yellow_{PDF} - 1.15 \times l, r - Red_{PDF} \quad (16)$$

Each of the stages-1, 2, 3, 4, 5 which are discussed in detail above are built and trained(only stage-2, 3) separately. The details of training methods, loss functions, hyperparameters used are discussed above at each respective stage's description. Training, testing and inferencing[5] of our proposed system is done on *Intel i7 9th-gen CPU coupled with Nvidia GTX-1070 GPU*. Different Training methods used above include, 1) Transfer learninig of imagenet pretrained [38] F-RCNN+Resnet-101_FPN [35], [36], [37] on our custom-built dataset along with other traffic datasets (Udacity, KITTI [23], [24]); the ensembled training data consists of O=7-object classes (Table 2). 2) In Stage-3, we have implemented 2-phase training methodology for the DSED network to learn & predict on our custom-built & CUlane datasets [21]. Custom(L_{Total})+ Pre-built Loss functions (L_{CE}, $L_{MSE,SE}$, L_{p-t}, L_δ) were used during training to optimize these networks using optimizers (Adam, SGD) [48]. Optimal hyper-parameters for stage-2, 3 were picked manually by checking performance of each tweaked parameter values calculated using Bayesian hyper-parameter optimization technique. Stage-1,4 were built completely using algorithms (ransac [30], DoG [25], HCD [26] etc) and CV techniques(blending [33], parameter calculations, mask generation & processing [28] etc). In Stage-5 a dense depth map is generated using a self supervising network (Monodepth2 [53]) to generate navigation suggestions. These separately built stages are executed concurrently according to the proposed Fig. 3 pipeline during inference phase, so that the proposed AMMDAS can dynamically generate robust real-time proximity alerts of objects identified within adaptive FOV range (θ^0) along with corresponding navigation suggestions (Tables 4 and 5) to ensure safety of the end-user.

[5]Memory space occupied by our proposed method during on-live inference is <1.4 GB this includes loading all the required meta-data, DNN weight files, algorithm pre-inputs, storing previously calculated internal parameter values etc.

V. EXPERIMENTS, RESULTS & DISCUSSION

We have tested and experimented the proposed system in live traffic recordings, where an initial setup of 2 smart-phones or web-cams are mounted to a stand separated by 120cm distance, the left and right stereo feeds are processed by stages 1-5 to output corresponding dynamic proximity alerts along with relative adaptive navigation suggestions (Tables 4 and 5). To evaluate the generalization and robustness of the proposed DAS we have additionally experimented on KITTI [24], Udacity[6] datasets (Table 5). As discussed above our custom-built dataset consists of 200+ minutes(110Kms) live traffic recordings, with a total of 2,67,300+ left-right video frames (1920*1080 resolution) extracted from the live traffic recordings. Each stereo pair is taken (Fig. 16(a), (b)) and passed to stage-1 to output 165^0 panoramic views (Fig. 16(e), (f)). Stage-1 achieved a throughput of 14fps during live traffic inferences, and detailed implementation, methods followed for generating a panoramic image are discussed above in section 4.A. In our custom dataset, 2,67,300 left-right stereo pairs yielded 2,53,980 panoramic images, the drop in image count between input l, r images and output panoramic images is due to the incapacity of DoG [25]+HCD [26] algorithm to detect relevant features (Fig. 16(c), (d)) between the l, r images to perform feature matching during wrapping operation [32], as the features weren't mapped to P.T projection plane the entire panoramic view construction for that respective pair will be discarded and skipped to next l, r pair. As the failure rate is <5% this incapacity wouldn't hinder the performance of the overall DAS module.

Generated panoramic views are annotated respectively for the training of stage-2, 3. The objects (O∈ 7-classes) present

[6]Open sourced at "https://github.com/udacity/self-driving-car"

TABLE 2. The 1st, 2nd, 3rd columns consist of object classes and their respective labels in the training data of stage-2, along with their respective segmentation colors in RGB format. Our custom-built data and KITTI [23], [24], Udacity datasets are labeled according to 1st, 2nd, 3rd formats, and FRCNN+Resnet-101_FPN network [35], [36] is trained on this data. 4th and 5th columns contain lane classes and their respective labels used in D.F.O.V.M training data, with their respective (R, G, B) segmentation colors. Our custom-built data + CU-Lane data [21] are labeled according to this 4th, 5th columns format (Fig. 7) and the DSED network is trained on this data.

J^{th}-Object's G.T	(R,G,B) color	J^{th}-Object's class or category	J^{th}-Lane's G.T color	J^{th} -Lane class in our Dataset
(0,0,255)	Blue	person/person-objects	White	right lane
(0,255,0)	Green	2-wheeler	Green	immediate right lane
(255,0,0)	Red	car	Olive green	immediate left lane
(0,255,255)	Cyan	non-terrestrial vehicles	Red	left lane
(255,255,0)	Yellow	heavy-vehicles	Purplish	background etc.
(255,0,255)	Magenta	animals/birds		
(210,210,220)	L. Grey	public-property		

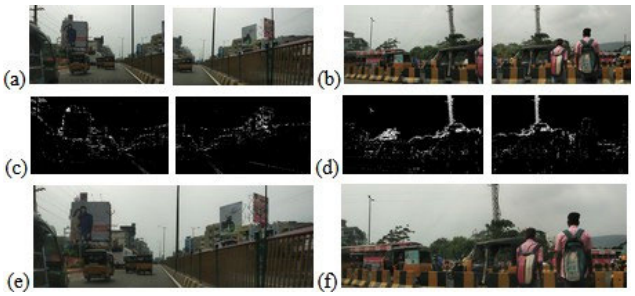

FIGURE 16. (a), (b) are input left, right stereo images, (c), (d) are intermediate feature descriptors identified using DoG+HCD algorithms on (a), (b) inputs respectively; (e), (f) are the final 165^0 panoramic view stitched stage-1's output (with exposure & artifacts compensated).

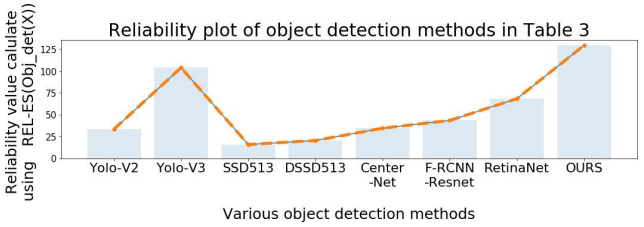

FIGURE 17. Reliability score plot(orange dashed line) of different object detection methods mentioned in Table 3. Reliability score is calculated using REL_ES function. The reliability-estimated-score is an inhouse determined metric(not used for generalized benchmarking), which generally indicates robustness and dependability of a specific object-detector when included in the STAGE-2 of AMMDAS.

in the front view's panoramic images and rear view's input frames are all together annotated based on the color-code of that respective object's class present (see Table 2) in the respective image. These annotated images are used to train the imagenet pre-trained FRCNN+ResNet-101_FPN network [35]–[37].

Similarly, 1,00,000 panoramic images are taken, and the lanes present in these images are annotated according to the color code (see Table 2) for that respective lane-class (Fig. 9(g), (h), (i)) for training the DSED network (Figs. 10 and 11), 5,000 images out of 1,00,000 are considered for testing and validation, 15,000 panoramic images(12K for training and 3K for testing, validation) are processed to convert the input panoramic image into Fig. 9(d), (e), (f), in which irrelevant surroundings are blurred and masked [28] to a black background for training mini-ED network (Fig. 10). In section 4.C (stage-3), 2-phase training method is implemented, in the 1st phase 15,000 annotated images are used to train the mini-ED of the DSED network, and in the 2nd phase, main-ED+ step-1 pre-trained mini-ED are together trained on 1,00,000 custom dataset +10K CU-Lane [21] +0.5K KITTI [22] images to output final lane segmented embedding image (Fig. 12(b)). Robust objects/l, c, r-FOV masks are constructed based on parameters/images generated by inferencing the trained stage-2, 3 networks (DSED & FRCNN+Resnet-101_FPN [35], [36]).

In section 4.B (Stage-2), FRCNN+Resnet-101_FPN network is chosen based on REL_ES score (Fig. 17) and performance comparison with other object detection networks (SSD [59], DSSD [60], Center-Net [61], ours, retinanet [62], Yolo [63]–[65]) during benchmarking and experimentation on [19] MS-$COCO_{test}$, our custom-traffic datasets. To measure the performance of each network (in Table 3) we took 5 parameters (4 parameters are detailed in Table 3, and 1 parameter in Fig. 17) into consideration mAP [36], FPS_I, FPS_O, TPR (true positive rate= #{objects detected by a respective method}/#{objects in ground truth image}), FPR(false positive rate= #{False positives detected by a respective method}/#{objects in ground truth image}) and REL_ES. mAP is mean average precision with 0.5-IoU-threshold on the COCO-2017 dataset [19] with 80 classes, FPS_I is ideal frames per second/throughput achieved by a method (stated in their paper) on respective test datasets (i.e, KITTI, PASCAL-VOC [20], COCO [19]), FPS_O^K is the overall throughput or inference time of our AMMDAS when K^{th}method is used in stage-2 for Object mask generation (Fig. 7). TPR, FPR are common evaluation metrics [41] to estimate robustness(TP) and accuracy(FP) of a particular method, here we have calculated the TPR, FPR of each method on a common test set containing 100 live frames of traffic recordings (custom dataset). REL_ES is reliability estimate score, which is calculated using i^{th}-object detector's mAP, FPS_O, TPR, FPR values, these values are passed as input to *REL(Obj(mAP, FPS, TPR, FPR))* function (*REL(obj)= ((mAP+TPR)*FPS_O)/FPR*) to estimate final reliability-estimate-score. Based on bench-marking results (Table 3, Fig. 17), Yolo-V3 [64]

TABLE 3. Performance of various object detectors on COCO dataset [19] with 0.5-IoU.

METHOD	[mAP]	FPS_I	FPS_O	TPR	FPR
Yolo-V2	48.1	40	6.4	69.7	11.3
Yolo-V3	55.3	34	6	90.1	4.2
SSD513	50.5	16	3.8	78.2	15.4
DSSD513	53.3	12	2.3	85.7	7.8
Center-Net	49.4	19	4.9	64.8	8.1
F-RCNN-Resnet	56.6	9	2.28	91.9	3.9
RetinaNet-101-500	59.2	5	0.95	97.7	1.09
Faster-RCNN+ Resnet-101_FPN	59.1	7	2.12	97.2	1.28

has the highest FPS when compared to all other methods and Retina-net [62], FRCNN+FPN-Resnet_101, (implemented method [35], [36]) have highest mAP, accuracies + robust-ness (high TPR, low FPR), [36], [41]. Retina-Net [62] has the least FPS when compared to other methods, making it infeasible for live application and detection. Our implemented method (FRCNN+Resnet-101_FPN) has ∼highest mAP, low-FPR, high-TPR scores [41] with greater $FPS_{I,O}$ when compared to Retina-Net [62] (refer to Table 3 for scores), making it possible to infer in a live environment, moreover our implemented method has the highest reliability score with a value of 129 followed by Yolo- V3 with a score of 102.8. Other methods Yolo-V2 [63], Center-Net [61] has low performance scores and SSD [59], DSSD [60], FRCNN [36], Yolo-V3 [64] have decent ac curacy scores and FPS values, but low TPR values of these methods indicate their inability to detect all available objects present in a surrounding, so this results in higher failure rate for triggering proximity alerts, difficulty and insufficiency in framing robust navigation suggestions as some necessary object parameters will be missed out during mutual analysis of l, c, r -BOL masks(stage-4). Fig. 18 illustrates SSD513 [59], Center-Net [61], YOLO-V3 [64] network's output object masks on some of the samples present in the benchmarking test-set. Center-Net [61] has least the TPR values followed by YOLO-V2 [63], as it is evident in Fig. 18(e), (k) that center-net [61] wasn't able to detect any objects which were present in the input FOV. SSD513 [59] has the highest FPR value as it produced the highest FP cases [41] compared to other methods.

In Fig. 18 (a), (d), (m), (p), SSD513 [59] misclassified detected objects. Moreover, Yolo-v3 [64] in Fig. 18(f), (l), (r) wasn't able to detect every object present in FOV but the detection results were better when compared to Center-Net [61], DSSD [60] and SSD513 [59], so YOLO-V3 [64] has low FPR and better TPR values. FRCNN+Resnet-101_FPN was able to detect every object present in the FOV and generated least FP cases [41], in Fig. 18(t), (v), (x) every object including persons on 2-wheelers (Fig. 18(u), (w)) and heavy vehicle (Fig. 18(s)), traffic warnings (Fig. 18(x)) etc were detected. SSD513 [59], Yolo-V-2, 3 [63], [64], Center-Net [61] misclassified a heavy-vehicle in Fig. 18(s) as a car, but Retina-net [62], DSSD [60], FRCNN+FPN(our OMM method) networks were able to classify accurately. Based

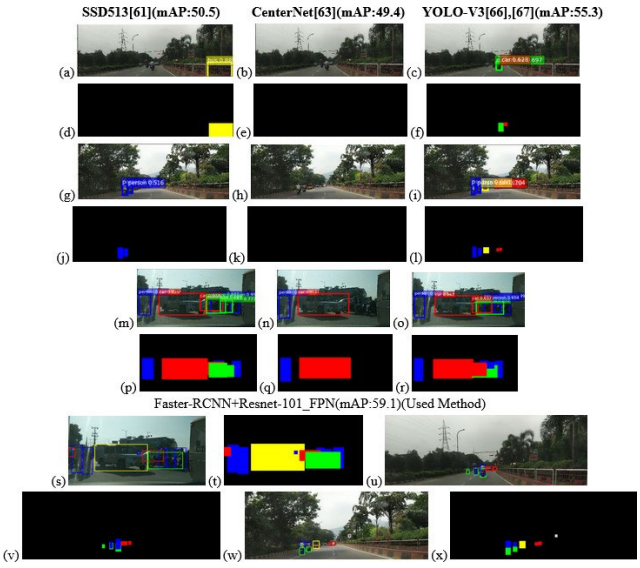

FIGURE 18. Object detection results of multiple object-dectectors on our sample test data, here we compare the results(refer to Table 3 for scores) of Center-net,YOLO-V3, SSD513, FRCNN+Resnet-101_FPN, (a), (g), (m) are object detection results of SSD513 with bounding boxes marked around the respective detected objects; and (d), (j), (p) are object masks generated by OMM_SSD513. (b), (h), (n) are object detection results of Center-Net, and (e),(k),(q) are object masks generated by OMM_Center-net; and (c), (i), (o) are Yolo-V3's object detection results and (f), (l), (r) are Object masks of OMM_YoloV3 module. (s), (u), (w) are object detection results by our implemented method(FRCNN+Resnet-101_FPN) and (t), (v), (x) are object masks generated by the proposed OMM module.

on the performance and reliability analysis with accuracy, throughput and REL_ES values as criteria from Table 3, Fig. 17 we were able to conclude that FRCNN+resnet-101_FPN network [35]–[37] was the most suitable to be included in Stage-2 (refer to section 4.B) for generating dynamic object masks.

DSED network (of D.F.O.V.M) implementation and training details are discussed in detail at section 4.C, the final output of the DSED network is a lane segmented embedding image and this output is further passed to lane point clustering algorithm [46] to generate respective lane clusters so that l,r parameters can be calculated to adaptively model the polygons present in l,r FOV masks (final D.F.O.V.M's output) [28], polygons present in center FOV masks are constant under any scenario (Fig. 21(i)). We evaluate the performance of stage-3 and DSED network by measuring the average of deviation between calculated l, r offset values (refer Fig. 20) and ground truth l, r offset values under different traffic/object-flow and lane visibility conditions. Different scenarios (Figs. 19 and 20) which were considered during D.F.O.V.M evaluation are 1: Low-Lane Visibility, 2: Moderate-Lane Visibility, 3: High-Lane Visibility, 4: Low-Traffic flow, 5: Moderate-Traffic flow, 6: High-traffic flow. The l, r deviation between predicted and ground truth is calculated on a total of 2500 test-set samples (Figs. 19 and 20) using (17). Images present in the test-set consider different

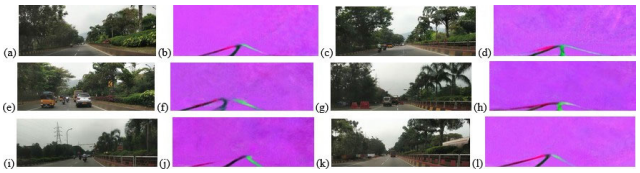

FIGURE 19. (a), (c), (e), (g), (i), (k) are different input scenarios given to the D.F.O.V.M module during performance analysis test, where each input image belongs to different scenarios i.e based on visibility of lane markings and object flow present in the outside environment. Different scenarios are detailed in Figure 22; and (b), (d), (f), (h), (j), (l) are the final lane segmented embedding images generated by the DSED network when respective input scenarios are fed to it. The lane segmentation in the final embedding images is performed according to the color codings (with respect to i^{th}-lane class) mentioned in Table 2.

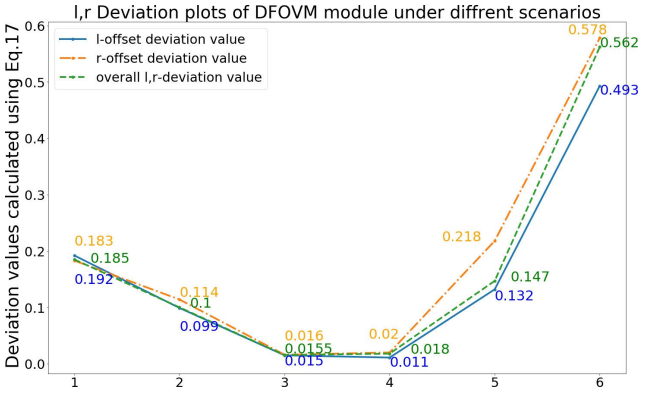

FIGURE 20. Performance benchmarking of the proposed method D.F.O.V.M under different external scenarios, where the scenarios are picked based on different lane visibility and object flow. Here we measure the deviations of left, right-offset values (Δ_N l(blue colored line), Δ_N r(orange colored line), Δ_N O(green colored line)) between DFOVM's predicted and ground truth values under these different scenarios.

levels(low moderate, high) of traffic flow and lane-visibilities for evaluation.

Where N = ∼400 images are sampled for each scenario, so that there wouldn't be any bias towards a specific scenario during Δ_Nl, Δ_Nr, Δ_NO calculation. Based on the above calculated values (Fig. 20) we can evaluate the robustness & accuracy of our AMMDAS in adapting to the outside surroundings under different scenarios. The feed from rear view isn't modeled because we follow a fixed FOV which will be followed throughout the execution in multiple stages, here fixed FOV refers to constant shape (Fig. 21(i), (j), (k)) of the polygons present in l, c, r FOV masks. Stage-3 performs very phenomenally well in High, moderate-Lane visibility & low, moderate-traffic flow conditions with a normalized deviation of less than 0.1 with respect to ground truth (refer to Fig. 20).

$$\begin{aligned}
\Delta_N^{l} &= \frac{1}{N} \times \sum_{i=0}^{N} \left(\Delta_{pred,GT}^{l_offset} \right) \\
\Delta_{pred,GT}^{l_offset} &= l_offset_{pred} - l_offset_{GT}; \\
\Delta_N^{r} &= \frac{1}{N} \times \sum_{i=0}^{N} \left(\Delta_{pred,GT}^{r_offset} \right) \\
\Delta_{pred,GT}^{r_offset} &= r_offset_{pred} - r_offset_{GT}; \\
\Delta_N^{O} &= \frac{1}{N} \sum_{i=0}^{N} \left(\frac{\Delta_i^{l} + \Delta_i^{r}}{2} \right)
\end{aligned} \tag{17}$$

Let's discuss the performance (Δ values are mentioned in Fig. 20 for different Figs. 19, 22 and 24 scenarios[7]) and outputs of D.F.O.V.M in Fig. 19 sample cases. Fig. 19(a), (k) have high lane line visibility and our DSED network generates lane segmentation maps (Fig. 19(b), (l)) close to ground truth in these cases and Fig. 19(i), (g), have moderate lane visibilities and even in these cases our DSED network generated lane segmentation maps (Fig. 19(j), (h)) close to ground truth and Fig. 19(c) have low-lane visibility and in this case, our DSED network outputs (Fig. 19(d)) an embedding image with slight distortions near lane segmentations. These distortions are caused due to the miss-classification of main-ED's L_{27} softmaxlayer in DSED, this misclassification happens due to dull lanes markings or due to large breaks in lanes, so feature maps extracted on these particular patches/image areas will be of low quality and these, low quality feature-maps raise FP cases [41] during L_{27} softmax classification leading to distortions and irregular lane thickness, patches in lane segments. These distortions and disturbing patches cause irregularity in lane clusters generated by the LPC algorithm therefore leading to small deviations in Δ values from G.T values. Fig. 19(a), (c), (g), (k) have low traffic flow and under these environments our DSED outputs lane segmented embedding image (Fig. 19(b), (d), (h), (l)) with highest IoU (the highest similarity to ground truth images) values with respect to ground truth images. In Fig. 19(e), (i) the traffic flow is moderate and in these cases our DSED sometimes generates embedding images (Fig. 19(d)) with slightly bloated lane segments leading to irregular thickness & structural orientations, this happens because objects and vehicles in FOV partially block the lane lines present on road leading to incomplete input lane patterns during prediction for lane segments in the final embedding image. When a lane is partially blocked by objects in input, our DSED estimates the blocked lane using an aggregated polynomial function approximated from lane patterns of ground truth annotated images (Fig. 9(g), (h), (i)), based on these estimated lane patterns, DSED segments the extracted lane patterns according to respective class's RGB color code (Table 2) to output final lane segmented embedding image. Incomplete Lane approximation can yield reliable outputs if at least 30% of the initial lane pattern is visible to the DSED network for prediction. Therefore, in these moderate traffic flow scenarios the deviation (Fig. 20, Δ_N O) rises from 0.01 to 0.15. In High traffic flow (Fig. 23(a), (c), Fig. 18(s)) cases the deviation value (Fig. 20) raises from 0.14 to 0.56 because the input lane lines are completely blocked(<30%), therefore the DSED network

[7]Diffrent scenarios in this very respective benchmarking analysis refer to 1:Low-Lane Visibility, 2:Moderate-Lane Visibility, 3:High-lane Visibility, 4:Low-traffic flow 5:Moderate-traffic flow, 6:High-traffic flow.

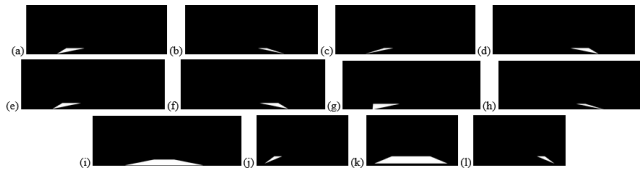

FIGURE 21. (a) to (i) are front view's adaptive left, center, right FOV masks; and (m), (n), (o) are rear-view's fixed left, center, and right FOV masks. (a), (c), (e), (g) are left FOV masks where the white-contour/polygon inside left-FOV mask is modeled according to the left-offset value estimated by the DSED+LPC algorithm [46], (i) is the center FOV mask and the contour/polygon inside center FOV remain constant under any situation. (b), (d), (f), (h) are right FOV masks where polygons inside r-FOV masks are modeled according to the r-offset value. Based on the Contours/polygons present inside l, c, r-FOV masks, FOV range ($\Theta_{FOV}{}^0$) in that particular scenario can be determined using (13). Generally, the range of $\Theta_{FOV}{}^0$ is in between 92^0-145^0.

cannot deduce any polynomial relations for lane patterns to estimate the incomplete lane lines in the final embedding image, so the output consists of irregular lane segmentation patterns with no proper correlations between them because L_{27}'s softmax layer receives final $L_{26,27}$ -feature-maps with no lane patterns extracted for prediction.

As discussed in section 4.C, l, r offset values are functions of both lanes, vehicles present in the external environment. So in high traffic flow scenarios the l, r offset parameters are majorly dependent on objects, vehicle density rather than lanes present outside, because LPC algorithm [46] tends to generate improper clusters and parameterization on these clusters leads to irregular semantic relations between the centroids. Therefore, Δ_N O value raises during high traffic flow as l, r offsets are calculated with major attention given towards external vehicles rather than on-road lanes, even in these cases, the generated l, c, r-FOVs were able to frame logical suggestions with reliable proximity triggers. Generally, in most of the high traffic flow cases, stage-4 includes an extra operational pipeline(as all l, c, r FOVs trigger proximity values above threshold) to execute stage-5 where a dense depth map [53] is generated to frame more robust navigation suggestions, thereby reducing failure and F.P cases [41]. Fig. 21(a)-(i) shows front view's l, c, r FOV masks respectively and Fig. 21(j), (k), (l) are fixed rear view's l, c, r FOV masks. White colored contours present in l, c, r FOV masks are the polygons which were modeled using respective l, r offset values. Fig. 21(a), (i), (b) are stage-3's output, when panoramic images corresponding to Fig. 19(k), (i) were given as input, here l, c, r offset parameters are 2, 1, 1 respectively, l-offset spans to a value of 2 and r-offset only to 1 because the host's input environment has more space towards left rather than the right side, so we model polygons accordingly in l, r FOV masks to cover more area towards left FOV so that more analysis [28] will be performed towards left to frame robust and logical suggestions and proximity alerts. Fig. 21(c), (i), (d) are outputs when inputs similar to Fig. 19 (c) are passed to stage-3, here l, c, r offset values are 1, 1, 2 (c-offset value remains constant(equals to 1) under any situation (Fig. 21(i))) respectively this is because spatial relief/space towards right side is more when compared to left. Fig. 21(e), (i), (f) are l, c, r-FOV masks respectively when Fig. 19(g) similar inputs are fed to stage-3, the l, c, r offset values are 2, 1, 2 respectively, here both l, r offset values are spanned to 2 because, left and right FOVs have an equal area which should be analyzed and both of the sides have an equal distribution of objects and spatial area so under these cases l, r offset values will be spanned equal. When stage-3 processes input scenarios which are in the nature of Fig. 19(a), (e) it outputs Fig. 21 (g), (i), (h) as l, c, r-FOV masks respectively based on $l = 3, c = 1, r = 1$ offset values, and l-offset value is 3 in this case because to the left there are 2 lanes (immediate left, left/pedestrian-service), so to analyze and process the entire left FOV we span the l-offset value to maximum(3) so that the estimated overall FOV ($\Theta_{FOV}{}^0$ calculated using (13)) adapts dynamically according to the host's external environment. FOV range may vary from 92^0-145^0 accordingly, minimum FOV range(92^0) when l, c, r offset values are 1, 1, 1 respectively, and when l, c, r are 3, 1, 3 then the system covers a maximum FOV range of 145^0.

The above generated front & rear view object and l, c, r-FOV masks are cumulatively passed as inputs to Mask post processing [28], [52] and logic decision module of stage-4, to dynamically output proximity alerts and adaptively frame robust navigation suggestions based on external conditions and environments. Initially stage-4 post-processes [28] the inputs (refer to section 4.B, 4.C) from previous stages to output l, c, r-BOL masks, so that PX_i^o parameters can be calculated(using (14)) to generate compounded l, c, r parameters for both front and rear views. These compounded parameters are mutually analyzed to output final proximity triggers and navigation suggestions. Detailed explanations of the steps involved in stage-4 are mentioned in section 4.D.

Here in this section we would like to cover different real time scenarios faced (Fig. 22 (a), (b), (c), (g), (h), (i), (j), (k), (l), (m)) while inferring on live traffic recordings along with performance & results((Fig. 22 (d), (e), (f), (n), (o), (p), (q), (r), (s), (t)) of the proposed system (AMMDAS) in these scenarios. Table 4, 1st column contains image_labels of multiple scenarios in live traffic & [21], [23], [24], [66], where each scenario contains either of the l, c, r FOV blocked by vehicles/objects, these scenarios in Table 4 are generalized, i.e during a live inference in real world traffic most of the input-frames, video-clip probably belongs to either one of these scenarios mentioned in Table 4. Table 4 2nd column consists of stage-3, 4 intermediate output parameters, which are necessary for framing suggestion and proximity triggers, 3rd column consists of final outputs. Each image present in Fig. 22(a), (b), (c), (g), (h), (i), (j), (k), (l), (m) resembles scenarios faced during a live traffic inference. Fig. 22(a), resembles a situation in which left & right FOVs are blocked by vehicles and center FOV consists of less object density compared to left, right FOVs. Inputs similar to Fig. 22(b) scenario has both right and center FOVs blocked by vehicles present on road and left FOV has comparatively less object concentration. Fig. 22(c), (i) both resemble left,

FIGURE 22. (a)-(t) consists of sample input panoramic images along with corresponding outputs which consists of framed proximity alerts and navigation suggestions. (a), (b), (c), (g), (h), (i), (m) are front view's panoramic input fed to the AMMDAS, simultaneously with (j), (k), (l) rear view's input feed. Here (g, j), (h, k), (i, l) pairs belongs to the same respective input instances. Adaptive left, center, right-FOV markings are drawn along with bounding boxes around the detected objects, and the AMMDAS's final output proximity alerts and corresponding navigation suggestions are also inscribed on these marked input panoramic images to simulate the final DAS output illustrations ((d), (e), (f), (n), (o), (p), (t)). Detailed description on the generated outputs is given in Table 4.

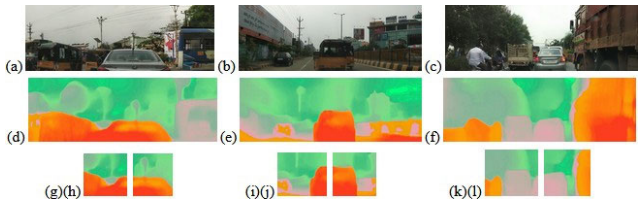

FIGURE 23. (a), (b), (c) are input images and (d), (e), (f) are corresponding HSV formatted depth maps generated by the Monodepth2 network [53]; [28] (red pixels represent the closest pixels and green represents the farthest pixels and the intermediary distant pixels are represented by orange and pink). (g), (h) are left, right FTC images generated for (a) input; (i), (j) are l, r-FTC images generated [52] for (b) input; and (k), (l) are l, r-FTC images generated for (c)-input. Based on histogram distributions and analysis on (g)-(l) l, r-FTC images Red, Green, Yellow$_{PDF}$ values are calculated and, using these PDF values left, right SRV values are determined.

right FOVs blocked and right FOV containing low object density scenario, but in Fig. 22(i) vehicles present in the host's FOV range have closer proximity when compared to vehicles in Fig. 22(c). Input feed (Fig. 22(a), (b), (c), (m)) does not contain any proximity alerts from rear view so their corresponding rear-view's feed is ignored (in Fig. 22), but Fig. 22(g), (h), (i) have proximity alerts from rear-view so their respective rear-view input feed is shown in Fig. 22. Fig. 22(g) has only left FOV blocked leaving right, center FOV comparatively free, and Fig. 22(h) has right FOV completely blocked and the other two FOVs have less object density compared to right FOV. Fig. 22(m) has proximity triggers only from center FOV, as the other two l, r-FOVs have lower object spatial concentration (i.e density) than the threshold (detailed section 4.B). Fig. 22(j), (k), (l) are input rear view's feed for respective front view(Fig. 22(g), (h), (i)) input panoramic feed. Fig. 22(j) has left FOV blocked and Fig. 22(k), (l) have both left & right FOVs blocked, in rear view the FOVs are inverted, where left (in image) implies right (in the real world) and vice versa this is because the front-view capturing cameras and rear view capturing cameras are placed in opposite directions (Fig. 1(a)) due to this the views in rear view images appeared to be inverted from real world view, and we output the proximity triggers+navigation suggestions with respect to real world's view/perception. Table 4 2nd column contains l, r-offset values which are stage-3's output for a respective 1st column input; based on these l,r offset values Θ^{fb}_{FOV} FOV-range(13) covered in that respective j^{th} 1st column scenario is also listed along with l, r offset values in 2nd column, Θ^{b}_{FOV} is constant (65^0) under any scenario but Θ^{f}_{FOV} is function of l, r-offset values and varies adaptively based on the input & host's external factors. Front/rear compounded l, c, r parameters (stage-4's intermediate outputs; techniques, algorithms [28], [52], steps involved in calculating these compounded l, c, r-parameters are detailed in section 4.D) of every 1st column's input scenario are also listed in 2nd column along with stage-3 outputs. These l, c, r-compounded parameters are key-value maps (dictionaries) with j^{th} object's (R, G, B) color code (which are in proximity) as keys and that respective j^{th} object's PX as values. Table 4's 3rd column contains(stage-4's final output) final proximity alerts for objects, vehicles which are in Θ_{FOV} range along with corresponding navigation suggestions for respective 1st column front+rear view input feed. In 3rd column, text under " **normal** " font indicates j^{th} object's proximity trigger in K-FOV and **text** in " **italic+bold"** indicate corresponding navigation suggestions which were framed by mutually analyzing respective parameters present in the 2nd column of Table 4.

These proximity alerts and navigation suggestions (stage-4's output) are robust and are reliable in most of the scenarios, but under some exceptions where the traffic flow is very high or the host vehicle's FOVs are completely blocked (all K-FOVs are blocked by surrounding objects; Fig. 23(a), (b), (c)) we include an additional Dense Depth Map generation and analysis module i.e stage-5 in the AM-MDAS inference pipeline. In Stage-5, input from stage-1 is converted to a dense depth (Fig. 15(b)) map by monodepth2 network [53]. We chose the Monodepth2 [53] network based on benchmarking ((15) F_β, RMSE) results (Table 1) with other monocular and stereo dense depth map generation modules [54]–[57]. This depth map is post-processed [52] to generate I^t (Fig. 23(d), (e), (f)) and from this I^t we extract left, right-FTC images (Fig. 23(g), (h), (i), (j), (k), (l), Fig. 15(d), (e)) so that left, right-SRV values can be calculated using R, G, B_{PDF} (using (16)) values of these l, r-FTC images(details are mentioned above in section 4.E) to frame final robust navigation suggestions to the end-user. Outputs from stage-5 contain only navigation suggestions and does not include any proximity triggers because stage- 5 is generally applied to inputs only when their

available FOVs are blocked by outside vehicles, so triggering proximity alerts in this case would lead to redundancy in output alerts and cluttred navigation suggestions to the end user. Outputs/suggestions from stage-5 start with "*All FOVs are blocked!! Suggesting based on a calculated depth map*" followed by a navigation suggestion (suggesting either to be towards left or right). Fig. 23(a) yields Fig. 23(d) as HSV post processed [28] dense depth map when fed to Monodepth2 network [53], and Fig. 23(g), (h) are corresponding left, right FTC images and, here in this case the left-SRV value is greater than right-SRV value, moreover it is evident that Fig. 23(h) has less Red_{PDF}, Yellow_{PDF} when compared to Fig. 23(g), so the final suggestion in this case would be "**All FOVs are blocked!! Suggesting based on a calculated depth map** *Better to be towards right*". For Fig. 23(b), (c) the resulting output HSV format depth maps [28] are Fig. 23(e), (f), now from these Depth maps respective left, right FTC images (Fig. 23(i), (j), (k), (l)) are processed [52] and extracted, in both of the above cases(b, c) left-SRV value is greater than right-SRV value, because Fig. 23(i), (k) have greater Green_{PDF} values and lower Yellow_{PDF}, Red_{PDF} values when compared to Red, Yellow, Green_{PDF} values of Fig. 23(j), (l); generally upon manually analysis of Fig. 23(b), (c) it's clearly evident that left side FOVs are more clear and have less object densities when compared to that of right side FOVs, So in these cases stage-5 outputs "**All FOVs are blocked!! Suggesting based on a calculated depth map** $\rightarrow$ *Better to be towards left*". The final outputs(Fig. 22(d), (e), (f), (n)-(t)) from Stage-4 & stage-5 are given to the end users in an interactive way, this output interactivity is modular which includes display on input panoramic feed, voice alerting, predefined-vibrating formats FPGA probing etc. In this paper we follow display on image format, where the outputs from stage-4, 5 are written on stage-1's panoramic image which is constructed from l, r input stereo feed.

Further discussion and analysis would be based on outputs and performance of the proposed system on different public datasets (KITTI [24] and Udacity [66]). Fig. 24(a), (b), (c), (d) and Fig. 24(i), (j), (k), (o) are input feed given to the proposed system and Table 5 contains the details and outputs for these Fig. 24 inputs. 1st, 3rd columns in Table 5 contain inputs from both Udacity and KITTI dataset [24], [66]. 2nd, 4th columns of Table 5 detail about the outputs (i.e proximity alerts along with corresponding navigation suggestions) generated from either stage-4 or stage-5 for a corresponding 1st, 3rd columns input. Moreover, 1st, 3rd columns also include respective Fig. 24 input's stage-3 internal calculated parameters (l, c, r-offset values, Θ_{FOV}) Based on the qualitative results from Tables 4 and 5 the proposed system/method was able to generate robust navigation suggestions by adaptively adjusting its FOV based on the external environment and also we were able to trigger proximity alerts with very low FP cases and high TP rate [41]. We have applied Bayesian optimization [67] on thresholding parameters in stage-2, 3 with an objective function to minimize the overall FPR value and to

FIGURE 24. Input Udacity [66]/KITTI [24] dataset images and their respective outputs along with corresponding proximity alerts and navigation suggestions. (a), (b), (c), (d), (i), (j), (k), (o) are KITTI/Udacity dataset inputs fed to the AMMDAS. Adaptive left, center, right-FOV markings are drawn along with bounding boxes around detected objects, simultaneously the AMMDAS's final output proximity alerts and corresponding navigation suggestions are inscribed on these marked input panoramic images to simulate the final DAS output illustrations (e), (f), (g), (h), (l), (m), (n), (p). (a), (b), (c), (d) inputs belong to the Udacity road dataset; and (i), (j), (k), (o) belong to the KITTI dataset. Detailed description on the generated outputs is given in Table 5.

increase TPR value [41] on the sample test dataset containing ensemble of all the above used data, later we manually tweaked threshold parameters in stage-4, 5 so that the final output contains generalized robust navigation suggestions. Fig. 24(a), (d) are underexposed and dark, Fig. 24(o) is over exposed to light and some images contain heavy shadows of respective objects/vehicles present outside(Fig. 24(a), (j)); even in these irregular lighting conditions and wild environments our proposed stages were able to generate Object masks, FOV masks and depth maps accurately and reliable, and the proposed pipeline was able to calculate internal parameters with robustness and minimal loss and, the resulting output to end user was very reliable.

In Fig. 25, the performance [41] of proposed AMMDAS "with" and "without " crucial proposed methods & features are measured for different scenarios. Different scenarios in this performance analysis refer to different lane visibility and traffic flow situations, different scenarios (x-axis) referred these graphs are *1: Low-Lane Visibility, 2: Moderate-Lane Visibility, 3: high-Lane Visibility, 4: Low Traffic-flow, 5: Moderate traffic-flow, 6: High traffic-flow, 7: wild/Noisy conditions*. These plots in Fig. 25 show the benefits and performance gain (in terms of TPR and FPR values [41]) achieved by including our proposed methods, features in a system. These proposed crucial methods, features are the functionalities that bring uniqueness and robustness in performance to our proposed AMMDAS. These features were comprehended and included upon brainstorming disadvantages and back-drops involved in other relevant DAS & ADAS systems.

The above mentioned crucial methods, feature are 1) processing wide panoramic feed (Fig. 25 top-left) which is generated by stitching stereo left, right cameras

TABLE 4. Description of the proposed DAS's outputs (proximity alerts with navigation suggestions) along with intermediary stage wise internal outputs i.e parameter values calculated in stage-3, 4 for respective Figure 22 inputs. 1st Column contains Figure 22 inputs, and 2nd Column contains stage-3, 4 internal calculated parameters (l, r-offset values, Front rear views l, c, r-compounded parameters) and 3rd Column contains the final proximity alerts along with corresponding navigation suggestions for respective 1st column inputs.

Inputs belonging to different scenarios	**l-offset, r-offset,** θ^{0}_{FOV}	**F/B Compounded l,c,r-Parameters wrt** j^{th}**- obj (R,G,B)color code**	**Prox. alerts & navigation. suggestions (output)**
Fig. 22(**a**)	{l: 1, c: 1, r: 2} Θ_{FOV}=107.5^{0}	{'0,0,255': 0.5901} {'0,0,255': 0.0848} {'0,255,0': 0.8718}	**No-alerts from rear-view **Danger! slow-down!! person or person-objects in left** → ***Better to be towards center*** Alert!! person or person-objects in center **Danger! slow-down!! 2-wheeler in right** → ***Better to be towards center***
Fig. 22(**b**)	{l: 2, c:1,r: 2} Θ_{FOV}=123^{0}	{} {'255,0,0': 0.7168} {'255,0,0': 1.0645}	**No-alerts from rear-view Nothing in left!! **Danger! slow-down!! car in center** → ***Better to be towards left*** **Danger! slow-down!! car in right** → ***Better to be towards left***
Fig. 22(**c**)	{l: 2, c:1,r: 2} Θ_{FOV}=123^{0}	{'255,0,0': 0.1481} {'255,0,0': 0.1579} {}	**No-alerts from rear-view Nothing in right!! Alert!! car in left Alert!! car in center
Fig. 22(**g**),(**j**) d∈F-view i∈B-view	{l: 1, c: 1, r: 1} $\Theta^{f}{}_{FOV}$=92^{0} $\Theta^{b}{}_{FOV}$=65^{0}	{'255,0,0':0.6737}{}{} {'0,0,255':0.3517} {'0,0,255':0.0529} {}	**Danger! car **behind** in left** Nothing in right!! **Danger! slow-down!! person or person-objects in left** → ***Better to be towards right*** Alert!! person or person-objects in center
Fig. 22(**h**),(**k**) e∈F-view j∈B-view	{l: 1, c: 1, r: 2} $\Theta^{f}{}_{FOV}$=107.5^{0} $\Theta^{b}{}_{FOV}$=65^{0}	{'0,255,0':0.8897}{} {'255,0,0':0.6901} {} {'255,255,0':0.2428} {'255,255,0': 1.1148}	**Danger! 2-wheeler **behind** in left** **Danger! car **behind** in right** Nothing in left!! **Danger! slow-down!! heavy-vehicles in center** → ***Better to be towards left*** **Danger! slow-down!! heavy-vehicles in right** → ***Better to be towards left***
Fig. 22(**i**),(**l**) f∈F-view k∈B-view	{l: 2, c:1, r: 2} $\Theta^{f}{}_{FOV}$=123^{0} $\Theta^{b}{}_{FOV}$=65^{0}	{'255,255,0':0.4122}{} {'0,255,0':0.9312} {'0,255,0':0.4129, '255,0,0': 0.4056} {'255,0,0': 0.3091} {}	**Danger! heavy-vehicles **behind** in left ** **Danger! 2-wheeler **behind** in right** Nothing in right!! **Danger! slow-down!! 2-wheeler in left** → ***Better to be towards right*** **Danger! slow-down!! car in left** → ***Better to be towards right*** **Danger! slow-down!! car in center** → ***Better to be towards right***
Fig. 22(**t**)	{l: 2, c: 1, r: 1} $\Theta^{f}{}_{FOV}$=123^{0} $\Theta^{b}{}_{FOV}$=65^{0}	{}{'255,0,255': 0.1551}{}	**No-alerts from rear-view Nothing in left!! Nothing in right!! Alert!! animals/birds in center

feed, 2) Implementing Adaptive field of view modeling (Fig. 25 top-right), 3) Mutually analyzing all the left, center, right FOVs (Fig. 25 bottom-left) in both front and rear view to generate intuitive proximity triggers and navigation suggestions and 4) Not restricting FOV range during input and processing + advantage of including monocular dense depth analysis during ambiguous situations (Fig. 25 bottom-right). Fig. 25 performance analysis is conducted on a testset with 6200 images, this test-set contains ∼1000 images per each scenario(1-6) and ∼230 images for 7^{th} scenario (wild/noisy conditions) only. TPR, FPR values for ''with'' and ''without'' systems are calculated according to the equation mentioned in Fig. 25 description. Generally ''with'' (orange, brown plotted lines, refer to each plot's legend in Fig. 25) plots are ground truth, i.e our proposed AMMDAS's performance plot in these different 1-7 scenarios (x-axis). In Fig. 25 top-left plot, the TPR values decrease & FPR values [41] rise slightly for ''with-out'' system's plot under high & moderate traffic scenarios because the AMMDAS wasn't getting an insightful broader input view for processing to generate final proximity triggers+navigation suggestions. For Fig. 25 top-right plot, the FPR values for ''with-out'' plot raise under high traffic flow and low, moderate-lane visibility scenarios and, TPR values of the same system drops at low, moderate lane visibility and high traffic flow scenarios, because when a DAS system performs adaptively modeling

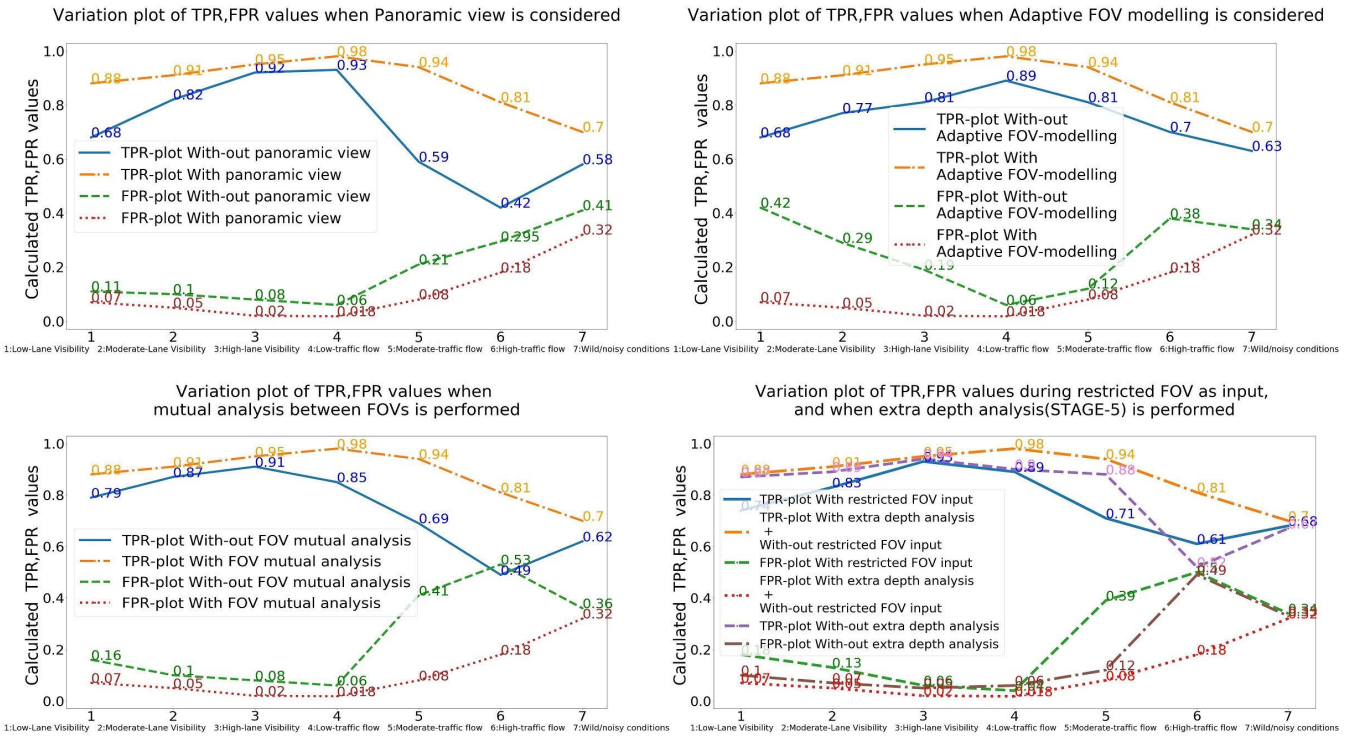

FIGURE 25. **Performance True-positive(TPR), False-positive-rate(FPR) plots of the AMMDAS with and with-out crucial methods/features which were introduced in this paper. Description and details of the plots are mentioned in legends, and in corresponding axis data. TPR and FPR values are calculated using [*TPR=#{True-positive [proximity triggers+navigation suggestions] framed on test dataset}/#{G.T proximity trigger and navigation suggestions present in test dataset}*], [*FPR= #{False-Positive [proximity triggers+navigation suggestions] framed on test dataset}/#{G.T proximity trigger and navigation suggestions present in test dataset}*].

FOV it can increase the chance of capturing and processing crucial objects, vehicles for generating final proximity alerts and suggestion, which are omitted in general scenarios and processing of these crucial objects increase the reliability and performance of a specific system. In Fig. 25 Bottom-left plot both TPR and FPR values [41] (of blue and green plotted lines) of ''with-out'' show higher fluctuations from ''with'' plot (orange and brown plotted lines) in moderate and high traffic flow scenarios because mutually analyzing each available FOV will increase the reliability of compounded l, c, r parameters calculated in stage-4, so that better logical navigation suggestions can be framed in the final outputs. Fig. 25 Bottom-right plot consists of ''with'' & ''with-out'' plots of both ''restricted FOV'' (blue and green line plots) and ''extra depth analysis'' (violet and brown line plots) features, in this plot; G.T plots i.e ''with-out restricted-FOV''' and ''with extra depth analysis'' are plotted with orange and red dashed lines, in this case FPR and TPR plots [41] of ''restricted FOV'' deviate from GT plots in every scenario except high lane visibility, low-traffic flow scenarios, because larger input FOV can give the system more attributes, criterias, arguments during processing to frame internal parameters for generating broad insights on external surrounds to frame much more correlated logical suggestions; TPR, FPR plots of ''extra depth analysis'' deviate from GT only under high traffic flow scenarios, because during high traffic flow scenarios more objects, vehicles in external surroundings get involved while estimating front-rear view compound parameters. So mutual or calculated parameters in these cases generate inaccurate alerts and suggestions because we are considering 2D areas of vehicles during K-BOL masks generation and internal parameter calculation, moreover as discussed above l, r offset parameters show higher deviations (Fig. 20) from ground truth in high traffic flow scenarios, therefore we have modeled our AMMDAS to enter depth analysis mode (stage-5) automatically during these ambiguous situations. So from the above benchmarkings, discussions and results, it's evident that the proposed system(AMMDAS) generates promising results in most of the scenarios and live-conditions. AMMDAS is a cost-effective DAS as it requires input feed only from front & rear view cameras, with no additional requirement of costly sensors (LIDAR, Radar etc.), our DAS runs on live and generates very reliable proximity alerts and suggestions in an interactive way to ensure the safety of end user during navigation.

VI. LIMITATIONS AND FUTURE WORK

The Limitations of the proposed DAS are, we do not consider extreme left & right field of views (FOV beyond 165^0), generally to cover this extreme left and right fields of view we need to increase the FOV range to 360^0, by doing so extra computational cost is required, simultaneously increasing the overall latency time. This proposed DAS fails to produce accurate results in extreme wild and noisy scenarios,

TABLE 5. Description of the proposed AMMDAS's outputs (proximity alerts with navigation suggestions) on Public udacity [66] & KITTI datasets [24], [23](Figure 24 inputs).1st, 3rd columns contains Figure 24 inputs along with corresponding DFOVM's internally calculated parameters (l, c, r-offset, Θ_{FOV} values); and 2nd, 4th Columns contains final proximity alerts with navigation suggestions for respective 1st, 3rd columns inputs.

Inputs & DFOVM output parameters	Prox. alerts & navigation. suggestions	Inputs & DFOVM output parameters	Prox. alerts & navigation. suggestions
Fig. 24(**a**) {l: 2,c: 1,r: 2} Θ_{FOV}=123^0	Nothing in left!! **Danger! slow-down!! car in center** → ***Better to be towards left*** **Danger! slow-down!! car in right** → ***Better to be towards left***	Fig. 24(**b**) {l: 1,c: 1,r: 2} Θ_{FOV}=107.5^0	Nothing in left!! **Danger! slow-down!! car in center** → ***Better to be towards left*** **Danger! slow-down!! car in right** → ***Better to be towards left***
Fig. 24(**c**) {l: 2,c: 1,r: 2} Θ_{FOV}=123^0	**Danger! slow-down!! person or person-objects in center** → ***Better to be towards left*** Alert!! car in center Alert!! person or person-objects in right Alert!! car in right	Fig. 24(**o**) {l: 2,c: 1,r: 3} Θ FOV=133.6^0	Alert!! person or person-objects in left Alert!! public-property in left Alert!! person or person-objects in right **Danger! slow-down!! car in right** → ***Better to be towards center***
Fig. 24(**i**) {l: 2,c: 1,r: 2} Θ_{FOV}=123^0	**Danger! slow-down!! car in left** → ***Better to be towards center*** Alert!! car in center **Danger! slow-down!! car in right** → ***Better to be towards center***	Fig. 24(**d**) {l: 1,c: 1,r: 2} Θ FOV=107.5^0	**All FOVs are blocked!! Suggesting based on a calculated depth map** → ***Better to be towards right***
Fig. 24(**k**) {l: 1,c: 1,r: 1} Θ_{FOV}=92^0	Nothing in left!! Alert!! car in center **Danger! slow-down!! car in right** → ***Better to be towards left***	Fig. 24(**j**) {l: 1,c: 1,r: 2} Θ_{FOV}=107.5^0	Nothing in left!! Alert!! car in right

i.e during heavy rainfall, very dull lights (nighttime with no street lighting), irregular terrains with no roads, etc. Our DAS system requires minimal manual calibration at the very start to yield more accurate results. The framed suggestions are sometimes lengthy with redundant structural repetitions and irregular grammatical formations (we plan to include NLP modeling in the future work for better interactivity). Currently, we are not considering traffic-light status, warning or caution symbols, pot-holes, and breakages [17] present on the road as criteria for framing navigation suggestions. Moreover, we mainly focus on spatial relationships between objects present in the input feed, results can be more accurate and reliable if temporal analysis between objects is also included, but by doing so the latency time, computational costs of a system increases. Moreover, we don't alert the driver or user under the proximity of any upcoming road turnings, curvatures. Our current methodology considers only 2D (w, h) object coordinates during stage-2, so plan to implement object detection in 3D (l, w, h) coordinates [68], [69] for stage-2. We also plan to introduce Advanced polygon modeling in D.F.O.V.M by implementing curvatures in edges for polygons in l, c, r-FOV masks with respect to on road lane line curvatures for extra robustness and reliability. The current research work mainly focuses on proposed AMMDAS system architecture and details about proposed sub-modules and their respective implementations; generally, a complete end-to-end DAS/ADAS systems require an on-premise bare-metal deployment (i.e on custom-built processor & hardware units) during live inferences. So, our future work includes details about optimized MIMO-hardware-implementations [70] of our AMMDAS on custom hardware units.

VII. CONCLUSION

This paper introduced a DAS which adaptively models its FOVs according to external dynamic surroundings. Based on the object's proximity in the adaptive FOV (θ^0_{FOV}) range, proximity alerts along with corresponding navigation suggestions were triggered to the end-user. This paper implemented a multi-stage strategy where each stage was built individually, and were concurrently executed together according to the proposed pipeline during inference. This paper overcame some of the major challenges & feature incapacities faced by other DAS systems (refer to section 2, Fig. 25), and was able to generate robust alerts and navigation suggestions even during poor lane visibility, high traffic flow and wild conditions (outputs in Tables 4 and 5). Each Stage introduced in our proposed pipeline tries to solve issues faced by existing vision based DAS systems. Where the proposed stage-1 generates a 165^0 wide panoramic view to create broader input FOV perspective for the later stages to processes, and stage-2 generates object masks for a respective input view using FRCNN+Resnet-101_FPN network [35], [36] to accurately identify object and vehicles (and their types) which are present in the input FOV. D.F.O.V.M of our pipeline generates a lane-segmented embedding image using the proposed DSED network to adaptively model its FOV based on external lanes and surroundings. Stage-4 processes all the previously generated masks of both front and rear views to mutually analyze them, for generating robust and reliable proximity alerts along with corresponding navigation suggestions. Stage-5 is elective, and is operational only under special cases to guarantee extra safety to the end-user by generating reliable navigation suggestions during high traffic flow. Although significant challenges were

handled in this paper, there will be a never ending quest for improvements in any technology, therefore some of the future enhancements we plan to include are mentioned in Section 6.

ACKNOWLEDGMENT

This work is done under the scholarship of Visvesvaraya Ph.D. Scheme for Electronics and IT, Government of India.

REFERENCES

[1] J. J. Rolison, S. Regev, S. Moutari, and A. Feeney, "What are the factors that contribute to road accidents? An assessment of law enforcement views, ordinary drivers' opinions, and road accident records," *Accident Anal. Prevention*, vol. 115, pp. 11–24, Jun. 2018, doi: 10.1016/j.aap.2018.02.025.

[2] M. Q. Khan and S. Lee, "A comprehensive survey of driving monitoring and assistance systems," *Sensors*, vol. 19, no. 11, p. 2574, Jun. 2019, doi: 10.3390/s19112574.

[3] A. Ess, K. Schindler, B. Leibe, and L. Van Gool, "Object detection and tracking for autonomous navigation in dynamic environments," *Int. J. Robot. Res.*, vol. 29, no. 14, pp. 1707–1725, Dec. 2010, doi: 10.1177/0278364910365417.

[4] S. Mittal, T. Prasad, S. Saurabh, X. Fan, and H. Shin, "Pedestrian detection and tracking using deformable part models and Kalman filtering," in *Proc. Int. SoC Design Conf. (ISOCC)*, Jeju Island, South Korea, vol. 10, Nov. 2012, pp. 324–327, doi: 10.1109/ISOCC.2012.6407106.

[5] D. Perrone, L. Iocchi, and P. C. Antonello, "Real-time stereo vision obstacle detection for automotive safety application," *IFAC Proc. Volumes*, vol. 43, no. 16, pp. 240–245, 2010, doi: 10.3182/20100906-3-IT-2019.00043.

[6] H. G. Jung, Y. H. Lee, B. J. Kim, P. J. Yoo, and J. H. Kim, "Stereo vision-based forward obstacle detection," *Int. J. Automot. Technol.*, vol. 8, no. 4, pp. 493–504, 2007.

[7] N. Ventroux, R. Schmit, F. Pasquet, P.-E. Viel, and S. Guyetant, "Stereovision-based 3D obstacle detection for automotive safety driving assistance," in *Proc. 12th Int. IEEE Conf. Intell. Transp. Syst.*, St. Louis, MO, USA, Oct. 2009, pp. 1–6, doi: 10.1109/ITSC.2009.5309832.

[8] V. K. Kukkala, J. Tunnell, S. Pasricha, and T. Bradley, "Advanced driver-assistance systems: A path toward autonomous vehicles," *IEEE Consum. Electron. Mag.*, vol. 7, no. 5, pp. 18–25, Sep. 2018, doi: 10.1109/MCE.2018.2828440.

[9] J. Cesićá, I. Marković, I. Cvisić, and I. Petrović, "Radar and stereo vision fusion for multitarget tracking on the special Euclidean group," *Robot. Auto. Syst.*, vol. 83, pp. 338–348, Sep. 2016, doi: 10.1016/j.robot.2016.05.001.

[10] S. Wu, S. Decker, P. Chang, T. Camus, and J. Eledath, "Collision sensing by stereo vision and radar sensor fusion," *IEEE Trans. Intell. Transp. Syst.*, vol. 10, no. 4, pp. 404–409, Dec. 2008, doi: 10.1109/TITS.2009.2032769.

[11] A. Howard, L. H. Matthies, A. Huertas, M. Bajracharya, and A. Rankin, "Detecting pedestrians with stereo vision: Safe operation of autonomous ground vehicles in dynamic environments," in *Proc. 13th Int. Symp. Robot. Res.*, Pasadena, CA, USA, 2007, p. 27.

[12] Z. Wang, W. Ren, and Q. Qiu, "LaneNet: Real-time lane detection networks for autonomous driving," 2018, *arXiv:1807.01726*. [Online]. Available: https://arxiv.org/abs/1807.01726

[13] A. Jazayeri, H. Cai, J. Y. Zheng, and M. Tuceryan, "Vehicle detection and tracking in car video based on motion model," *IEEE Trans. Intell. Transp. Syst.*, vol. 12, no. 2, pp. 583–595, Jun. 2011, doi: 10.1109/tits.2011.2113340.

[14] L. Qian, X. Qiwei, S. Mita, K. Ishimaru, and N. Shirai, "Small object detection based on stereo vision," *Int. J. Automot. Eng.*, vol. 7, no. 1, pp. 9–14, 2016, doi: 10.20485/jsaeijae.7.1_9.

[15] B. Wang, S. A. R. Florez, and V. Fremont, "Multiple obstacle detection and tracking using stereo vision: Application and analysis," in *Proc. 13th Int. Conf. Control Autom. Robot. Vis. (ICARCV)*, Singapore, Dec. 2014, pp. 1074–1079, doi: 10.1109/ICARCV.2014.7064455.

[16] A. Seki and M. Okutomi, "Robust obstacle detection in general road environment based on road extraction and pose estimation," *Electron. Commun. Jpn. (Part II, Electron.)*, vol. 90, no. 12, pp. 12–22, 2007, doi: 10.1002/ecjb.20413.

[17] F. Oniga and S. Nedevschi, "Processing dense stereo data using elevation maps: Road surface, traffic isle, and obstacle detection," *IEEE Trans. Veh. Technol.*, vol. 59, no. 3, pp. 1172–1182, Mar. 2010, doi: 10.1109/tvt.2009.2039718.

[18] A. Broggi, M. Buzzoni, M. Felisa, and P. Zani, "Stereo obstacle detection in challenging environments: The VIAC experience," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, San Francisco, CA, USA, Sep. 2011, pp. 1599–1604, doi: 10.1109/IROS.2011.6094535.

[19] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Feb. 2015, pp. 1–15. [Online]. Available: http://arXiv:1405.0312v3

[20] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal visual object classes (VOC) challenge," *Int. J. Comput. Vis.*, vol. 88, no. 2, pp. 303–338, Jun. 2010, doi: 10.1007/s11263-009-0275-4.

[21] X. Pan, J. Shi, P. Luo, X. Wang, and X. Tang, "Spatial as deep: Spatial CNN for traffic scene understanding," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.* Menlo Park, CA, USA: Association Advancement Artificial Intelligence, Dec. 2017, [Online]. Available: http://www.aaai.org and http://arXiv:1712.06080v1

[22] J. Fritsch, T. Kuhnl, and A. Geiger, "A new performance measure and evaluation benchmark for road detection algorithms," in *Proc. 16th Int. IEEE Conf. Intell. Transp. Syst. (ITSC)*, Oct. 2013, pp. 1693–1700, doi: 10.1109/ITSC.2013.6728473.

[23] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.* Los Alamitos, CA, USA: IEEE Computer Society, Jun. 2012, pp. 3354–3361. [Online]. Available: http://ieeecomputersociety.org/10.1109/CVPR.2012.6248074, doi: 10.1109/cvpr.2012.6248074.

[24] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, Sep. 2013, doi: 10.1177/0278364913491297.

[25] L. Assirati, N. R. Silva, L. Berton, A. A. Lopes, and O. M. Bruno, "Performing edge detection by difference of Gaussians using q-Gaussian kernels," *J. Phys., Conf. Ser.*, vol. 490, Mar. 2014, Art. no. 012020, doi: 10.1088/1742-6596/490/1/012020.

[26] G. Vino and A. D. Sappa, "Revisiting Harris corner detector algorithm: A gradual thresholding approach," in *Image Analysis and Recognition*, vol. 7950, A. M. Kamel and A. Campilho, Eds. New York, NY, USA: Springer-Verlag, 2013, pp. 354–363.

[27] J. N. Surekha, Y. S. Chakrapani, and N. V. Rao, "A survey of trends in local invariant feature detectors," *Int. J. Elect. Electron. Eng. Telcommun.*, vol. 6, no. 2, pp. 9–17, 2017.

[28] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*, 3rd ed. Englewood, Cliffs, NJ, USA: Prentice-Hall, Mar. 2009, doi: 10.1117/1.3115362.

[29] J. Hosang, R. Benenson, and B. Schiele, "Learning non-maximum suppression," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Honolulu, HI, USA, May 2017, pp. 6469–6477. [Online]. Available: http:/arXiv:1705.02950, doi: 10.1109/CVPR.2017.685.

[30] U. Baid, "Image registration and homography estimation," Shri Guru Gobind Singhji Inst. Eng. Technol., Nanded, India, Tech. Rep., 2015, doi: 10.13140/RG.2.2.19709.67043.

[31] R. Panigrahy, "An improved algorithm finding nearest neighbor using Kd-trees," in *LATIN 2008: Theoretical Informatics*, vol. 4957. New York, NY, USA: Springer-Verlag, 2008, pp. 387–398.

[32] L. G. Brown, "A survey of image registration techniques," *ACM Comput. Surv.*, vol. 24, no. 4, pp. 325–376, Dec. 1992, doi: 10.1145/146370.146374.

[33] P. Pérez, M. Gangnet, and A. Blake, "Poisson image editing," *ACM Trans. Graph.*, vol. 22, no. 3, p. 313, 2003. 10.1145/882262.882269.

[34] M. Wahab, "Interpolation and extrapolation," in *Proc. Topics Syst. Eng. Winter Term*, vol. 17, Jan. 2017, pp. 1–6.

[35] T.-Y. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Honolulu, HI, USA, Jul. 2017, pp. 936–944, doi: 10.1109/CVPR.2017.106.

[36] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal net-works," in *Proc. NIPS*, 2015, pp. 91–99. [Online]. Available: http://papers.nips.cc/paper/5638-faster-r-cnn-towards-real-time-object-detection-with-region-proposal-networks.pdf

[37] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Dec. 2015, pp. 1–12. [Online]. Available: http://arXiv:1512.03385v1

[38] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet large scale visual recognition challenge," *Int. J. Comput. Vis.*, vol. 115, no. 3, pp. 211–252, Dec. 2015, doi: 10.1007/s11263-015-0816-y.

[39] V. Dumoulin and F. Visin, "A guide to convolution arithmetic for deep Learning," 2016, *arXiv:1603.07285*. [Online]. Available: https://arxiv.org/abs/1603.07285

[40] H. Wu and X. Gu, "Max-pooling dropout for regularization of convolutional neural networks," 2015, *arXiv:1512.01400*. [Online]. Available: https://arxiv.org/abs/1512.01400

[41] J. Davis and M. Goadrich, "The relationship between precision-recall and ROC curves," in *Proc. 23rd Int. Conf. Mach. Learn. (ICML)*. New York, NY, USA: Association Computing Machinery, 2006, pp. 233–240, doi: 10.1145/1143844.1143874.

[42] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proc. 27th Int. Conf. Mach. Learn. (ICML)*, 2010, pp. 807–814.

[43] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. 32nd Int. Conf. Int. Conf. Mach. Learn.*, vol. 37, Lille, France, Mar. 2015, pp. 448–456. [Online]. Available: http://arXiv:1502.03167v3.

[44] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2015, pp. 3431–3440.

[45] J. C. Ye and W. K. Sung, "Understanding geometry of encoder-decoder CNNs," 2019, *arXiv:1901.07647*. [Online]. Available: https://arxiv.org/abs/1901.07647

[46] J. Qi, Y. Yu, L. Wang, and J. Liu, "K*-means: An effective and efficient K-means clustering algorithm," in *Proc. IEEE Int. Conferences Big Data Cloud Comput. (BDCloud), Social Comput. Netw. (SocialCom), Sustain. Comput. Commun. (SustainCom) (BDCloud-SocialCom-SustainCom)*, Oct. 2016, pp. 242–249.

[47] H. Gao, H. Yuan, Z. Wang, and S. Ji, "Pixel deconvolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Nov. 2017, pp. 1–10. [Online]. Available: http://arXiv:1705.06820v4

[48] S. Sun, Z. Cao, H. Zhu, and J. Zhao, "A survey of optimization methods from a machine learning perspective," *IEEE Trans. Cybern.*, vol. 50, no. 8, pp. 3668–3681, Aug. 2020, doi: 10.1109/TCYB.2019.2950779.

[49] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 12, pp. 2481–2495, Dec. 2017, doi: 10.1109/tpami.2016.2644615.

[50] N. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Santiago, Chile, Dec. 2015, pp. 1520–1528. [Online]. Available: http://arXiv:1505.04366v1, doi: 10.1109/ICCV.2015.178.

[51] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," in *Medical Image Computing and Computer-Assisted Intervention—MICCAI 2015*, vol. 9351. Cham, Switzerland: Springer, 2015, pp. 234–241. [Online]. Available: https://arXiv:1505.04597v1, doi: 10.1007/978-3-319-24574-4_28.

[52] B. G. Batchelor and F. M. Waltz, "Morphological image processing," in *Machine Vision Handbook*, B. G. Batchelor, Ed. London, U.K.: Springer, 2012, pp. 801–870. [Online]. Available: https://doi.org/10.1007/978-1-84996-169-1.

[53] C. Godard, O. M. Aodha, M. Firman, and G. Brostow, "Digging into self-supervised monocular depth estimation," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Seoul, South Korea, Nov. 2019, pp. 3827–3837. [Online]. Available: http://arXiv:1806.01260v4, doi: 10.1109/ICCV.2019.00393.

[54] O. Godard, O. M. Aodha, and G. J. Bros-Tow, "Unsupervised monocular depth estimation with left-right consistency," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Honolulu, HI, USA, Jul. 2017, pp. 6602–6611. [Online]. Available: http://arXiv:1609.03677v3, doi: 10.1109/CVPR.2017.699.

[55] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe, "Unsupervised learning of depth and ego-motion from video," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Honolulu, HI, USA, Jul. 2017, pp. 6612–6619. [Online]. Available: http://arXiv:1704.07813v2, doi: 10.1109/CVPR.2017.700.

[56] K. Karsch, C. Liu, and S. B. Kang, "Depth transfer: Depth extraction from video using non-parametric sampling," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 11, pp. 2144–2158, Nov. 2014, doi: 10.1109/TPAMI.2014.2316835.

[57] C. Wang, J. M. Buenaposada, R. Zhu, and S. Lucey, "Learning Depth from Monocular Videos using Direct Methods," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Salt Lake City, UT, USA, Jun. 2018, pp. 2022–2030. [Online]. Available: http://arXiv:1712.00175v1, doi: 10.1109/CVPR.2018.00216.

[58] A. Saxena, M. Sun, and A. Y. Ng, "Make3D: Learning 3D scene structure from a single still image," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 5, pp. 824–840, May 2009, doi: 10.1109/TPAMI.2008.132.

[59] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *Computer Vision—ECCV 2016*, vol. 9905. Cham, Switzerland: Springer, 2016, pp. 21–37, doi: 10.1007/978-3-319-46448-0_2.

[60] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and C. Alexander, "DSSD: Deconvolutional single shot detector," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jan. 2017, pp. 1–11. [Online]. Available: http://arXiv:1701.06659v1

[61] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, "CenterNet: Keypoint triplets for object detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Seoul, South Korea, Nov. 2019, pp. 6568–6577. [Online]. Available: http://arXiv:1904.08189v3, doi: 10.1109/ICCV.2019.00667.

[62] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Venice, Italy, Oct. 2017, pp. 2999–3007. [Online]. Available: http://arXiv:1708.02002v2, doi: 10.1109/ICCV.2017.324.

[63] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Honolulu, HI, USA, Jul. 2017, pp. 6517–6525. [Online]. Available: http://arXiv:1612.08242v1, doi: 10.1109/CVPR.2017.690.

[64] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Apr. 2018, pp. 1–6. [Online]. Available: http://arXiv:1804.02767v1

[65] 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. (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 779–788. [Online]. Available: http://arXiv:1506.02640v5, doi: 10.1109/CVPR.2016.91.

[66] (2020). *Udacity, Udacity/Self-Driving-Car*. [Online]. Available: https://github.com/udacity/self-driving-carc

[67] J. Snoek, H. Larochelle, and R. P. Adams, "Practical Bayesian optimization of machine learning algorithms," in *Proc. 25th Int. Conf. Neural Inf. Process. Syst.*, vol. 2. Red Hook, NY, USA: Curran Associates, Dec. 2012, pp. 2951–2959.

[68] A. Osep, W. Mehner, M. Mathias, and B. Leibe, "Combined image- and world-space tracking in traffic scenes," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Singapore, May 2017, pp. 1988–1995, doi: 10.1109/ICRA.2017.7989230.

[69] A. Osep, A. Hermans, F. Engelmann, D. Klostermann, M. Mathias, and B. Leibe, "Multi-scale object candidates for generic object tracking in street scenes," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, Stockholm, Sweden, May 2016, pp. 3180–3187, doi: 10.1109/ICRA.2016.7487487.

[70] L. Ma, X. Huo, X. Zhao, and G. D. Zong, "Observer-based adaptive neural tracking control for output-constrained switched MIMO nonstrict-feedback nonlinear systems with unknown dead zone," *Nonlinear Dyn.*, vol. 99, no. 2, pp. 1019–1036, Jan. 2020.

VENKATA SUBBAIAH DESANAMUKULA received the B.Tech. degree from Nagarjuna University, and the M.Tech. degree from BI-HER, Chennai. He is currently pursuing the Ph.D. degree under Visvesvaraya Ph.D. Scheme, Government of India, with the Department of Computer Science and Systems Engineering, Andhra University, Visakhapatnam. His research interests include object detection using deep learning, cyber security, and machine learning.

PREMITH KUMAR CHILUKURI received the B.Tech. degree in computer science. He is currently pursuing research in deep-learning and image-processing domains at Andhra University College of Engineering (A). His research interests include autonomous systems, neural cognitive intelligence, computer vision, deep learning, reinforcement learning, and signal/image processing (with a current expertise of more than three years in these research areas).

PUSHKAL PADALA is actively participating in various real time applications of machine learning. His research interests include image processing, pattern recognition, and data security.

PREETHI PADALA is currently pursuing the B.Tech. degree with the National Institute of Technology Karnataka. Her research interests include artificial intelligence, image processing, pattern recognition, machine learning, and data sciences.

PRASAD REDDY PVGD is currently a Senior Professor with the Department of Computer Science and Systems Engineering, Andhra University College of Engineering (A), Andhra University. His research interests include machine learning, soft computing, software architectures, image processing, and number theory and cryptosystems.