A Stereo Perception Framework for Autonomous Vehicles

Narsimlu Kemsaram, Anweshan Das, and Gijs Dubbelman

***Abstract*— Stereo cameras are crucial sensors for self-driving vehicles as they are low-cost and can be used to estimate depth. It can be used for multiple purposes, such as object detection, depth estimation, semantic segmentation, etc. In this paper, we propose a stereo vision-based perception framework for autonomous vehicles. It uses three deep neural networks simultaneously to perform free-space detection, lane boundary detection, and object detection on image frames captured using the stereo camera. The depth of the detected objects from the vehicle is estimated from the disparity image computed using two stereo image frames from the stereo camera. The proposed stereo perception framework runs at 7.4 Hz on the Nvidia Drive PX 2 hardware platform, which further allows for its use in multi-sensor fusion for localization, mapping, and path planning by autonomous vehicle applications.**

Index Terms— advanced driver assistance system, autonomous vehicle, deep neural network, depth estimation, free space detection, lane detection, object detection, stereo camera, stereo perception, stereo vision.

I. Introduction

Perceiving the environment accurately in real-time is one of the most challenging tasks for autonomous vehicles. Perception refers to the ability of the autonomous vehicle to collect sensor data, extract relevant knowledge, and develop a contextual understanding of the environment, for example, detection of obstacles, lanes, and the drivable area in front of the vehicle [1]. Sensors such as cameras, lidars, and radars are used in autonomous driving vehicles to perceive the environment around it. LiDARs are very accurate active depth measurement sensors, but these are very expensive and are not ready to be equipped on consumer-grade vehicles. Radars are robust sensors used in advanced driver assistance systems (ADAS) like adaptive cruise control (ACC), blind-spot detection, etc. But the resolution of radar is not high enough to extract semantic information of the surrounding. Cameras, on the other hand, are the only sensor that is comparatively less expensive and provides a high level of details such as color, contrast, texture information, which allows for better semantic understanding of the environment and can also be used to estimate depth. With ever-increasing performance in dynamic lighting conditions and being relatively cheap to manufacture, camera-based systems are widely used in today's ADAS and autonomous vehicles. Monocular camera-based systems are generally used for ADAS applications such as ACC [2], forward collision warning (FCW) [3], lane departure warning (LDW) [4], etc.

Narsimlu Kemsaram (n.kemsaram@tue.nl), Anweshan Das (anweshan.das@tue.nl) and Gijs Dubbelman (g.dubbelman@tue.nl) are with the Department of Electrical Engineering, Mobile Perception Systems Research Lab, Video Coding and Architectures/Signal Processing Systems Research Group, Eindhoven University of Technology (TU/e), 5612 AZ Eindhoven, The Netherlands.

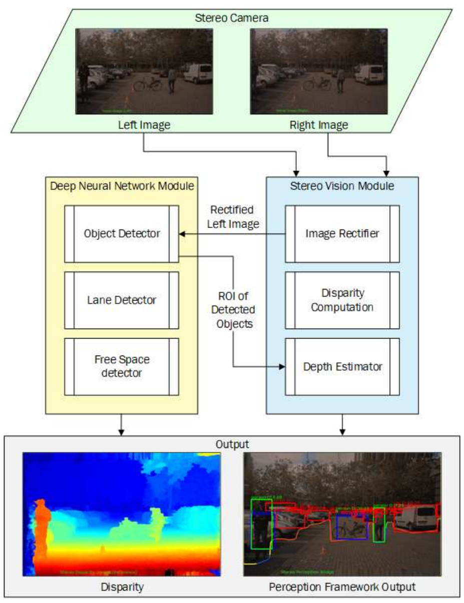


Fig. 1: Block Diagram of the Proposed Stereo Perception Framework.

In recent years there has been a lot of advancement in the field of deep learning. Deep neural networks (DNNs), specifically different variants of convolutional neural network (CNN) have been used in the field of computer vision for complex tasks like object detection [5], pattern recognition [6], segmentation [7], depth estimation [8], etc. It even outperforms human beings in tasks like object recognition and pattern recognition [9]. The main drawback of complex DNNs is that it is computationally very expensive and cannot meet real-time system constraints. But, with recent development in graphical processing units (GPUs) and DNN optimized hardware platforms, it is becoming easier to deploy DNNs for real-time applications.

Depth estimation of objects around the surrounding can be performed using a monocular camera or a stereo camera. There is no direct method to estimate depth from a monocular camera. The depth information of the surroundings can be estimated by assuming the surface to be planar or by using image feature tracking with additional information about the twist of the camera in subsequent frames [10]. DNN based monocular depth estimation techniques are not robust enough to be used in automotive-grade applications. Stereo cameras use intrinsic projective geometry between two views, it is independent of scene structure and only depends on camera's internal and external parameters. It is widely used for depth estimation in the robotics and automotive domain.

In this paper, we propose and evaluate a stereo camera-based perception framework for autonomous vehicles. Figure 1 shows the block diagram of the stereo perception framework. The framework uses DNNs to perform lane boundary detection, free space detection, object detection, and classification on the left image frame of the stereo camera. It uses the left and right image frame for the stereo camera to compute a disparity image then estimates the depth of the detected objects. The framework requires a lot of parallel processing power from GPUs. PCs with consumer-grade GPUs consume a lot of electricity, and vehicles do not produce that much of electrical energy. We use Nvidia's Drive PX 2 platform to deploy the framework [11]. The Drive PX 2 is a very powerful and efficient automotive-grade platform which can be used to deploy highly optimized DNNs for real-time applications.

The main contributions of this paper are: i) Developed and integrated a stereo camera-based perception framework for autonomous vehicles, ii) Developed depth estimation module, iii) Evaluated performance of the depth estimation and stereo perception framework in real-time.

This paper is structured as follows: Section II discusses about the camera calibration process. Section III provides details of the stereo vision module. Section IV provides details of the DNN module. Section V explains the proposed stereo perception framework in detail. Section VI explains the experimental setup in detail. Section VII discusses the experimental results. Section VIII evaluates the depth estimation and stereo perception framework performance. In the Section IX presents the paper conclusions.

II. Camera Calibration

The process of estimating the intrinsic and extrinsic parameters of a camera is called camera calibration [12]. Since the manufacturing process of camera sensors and its lenses are never perfect, precise camera calibration is essential to re-project 2D images into the 3D world. The intrinsic parameter characterize the geometric, digital, and optical characteristics of the camera. It is specific to each camera. It is composed of the principal point or optical center (c_x, c_y), the focal length (f_x, f_y), the pixel size (p_x, p_y), the skew coefficient (s), the camera image resolution, and the lens distortion coefficients (k_1, k_2, p_1, p_2). The extrinsic parameter represents the six degrees of freedom (6DoF) pose of the camera in the world. It is represented by a translation vector T and a rotation matrix R [13]. Stereo camera calibration is the process of estimating the intrinsic and extrinsic parameter of two cameras in the stereo setup.

We use the Matlab stereo calibration toolbox [14] to perform stereo calibration. It uses multiple images of a checkerboard pattern captured using the stereo camera to estimate the intrinsic and extrinsic parameters, for more details on how the toolbox works, please refer to [14].

The calibration toolbox computes the camera extrinsic parameters of the stereo camera, considering the left camera's optical center as the origin, which needs to be again transformed to the vehicle coordinate system using a rigid body transformation. The vehicle uses a right-handed coordinate system, where the vehicle origin is considered to be under the center of the rear axle. The x-axis points forward to the front of the vehicle, the y-axis points to the left of the vehicle, and the z-axis points to the upward of the vehicle. The camera has a right-handed coordinate system, where the camera origin is at the optical center of the left camera. The x-axis points to the right of the image plane, the y-axis points to the bottom of the image plane, and the z-axis points forward along the optical axis. The estimated intrinsic and transformed extrinsic parameters are written to a rig configuration file in XML format, which is used in the stereo perception framework.

III. Stereo Vision

The stereo vision module computes the depth of the detected objects in the stereo perception framework. It is divided into three parts, namely stereo rectification, disparity computation, and depth estimation.

A. Stereo Rectification

The left and right image frames of the stereo camera are undistorted and rectified before computing disparity. The left and right cameras are modeled as a pinhole camera [15]. Image undistortion refers to the process of removing lens distortion artifact form the image. Lens distortion is modeled as two types, namely radial distortion and tangential distortion. Straight lines in an image appear curved due to radial distortion, and the effect is more prominent as we move away from the center of the image. It is represented using two coefficients k_1, k_2. Tangential distortion occurs if the lens is not mounted parallel to the image sensor. It is represented using two coefficients p_1, p_2. Let's consider that a point in 3D when projected on a camera image plane is represented as (x_1, y_1), but due to lens distortion, it appears to be at (x_2, y_2). The distorted points are described using the following function [16]:

$$x_2 = x_1(1 + k_1 r^2 + k_2 r^4) + 2p_1 x_1 y_1 + p_2(r^2 + 2x_1^2) \quad (1)$$

$$y_2 = y_1(1 + k_1 r^2 + k_2 r^4) + 2p_2 x_1 y_1 + p_1(r^2 + 2y_1^2) \quad (2)$$

where, r is the distance of the pixel from the optical centre, $r = \sqrt{x_1^2 + y_1^2}$. These coefficients are estimated during the camera calibration process.

The undistorted left and right camera images are then rectified. Image rectification is a transformation process that projects left and right camera images onto a common plane parallel to the line between optical centers of the camera such that the epipolar lines become collinear and parallel to the horizontal image axes. In simple words, the image transformation is such the projection of every point in the left image will have a corresponding point on the right image on a horizontal line, which is collinear and parallel to the horizontal image axes. This limits the search space for stereo correspondence [17]. For more details on the image rectification process, refer to [18].

B. *Stereo Disparity*

The disparity of all pixels, also known as the disparity map, is computed using the stereo image pair. The disparity is the distance between two corresponding points in the left and right images of a stereo pair. The problem of finding pixel correspondences between a stereo image pair is called stereo matching. After image rectification, the search space for the corresponding pixel is constrained to the epipolar line. Pixels are matches by comparing the sum of absolute difference (SAD) or sum of squared differences (SSD) or normalized cross-correlation (NCC) of the intensity of pixels around it [19]. The disparity d of each pixel is represented as:

$$d = (u_l - u_r) \tag{3}$$

where, (u_l) and (u_r) is the horizontal positions of the correspondence pixel on the left and right image plane.

C. *Depth Estimation*

The output of the object detection and tracking module, which is explained in Section IV-A, is a bounding box around the detected object along with its label. The pixel disparity values of the detected objects are available from the computed disparity map. The depth of a pixel is estimated from its disparity value using the triangulation equation from the stereo geometry [20]:

$$z = f * b/d \tag{4}$$

where, z is the depth, f is a focal length, b is a baseline distance between the left and right camera, d is the disparity.

IV. Deep Neural Network

This part of the framework processes image frames to understand the surroundings of the autonomous vehicle. It consists of three modules, namely object detection and tracking, lane detection, and free space detection, which are explained in detail below.

A. *Object Detection and Tracking*

The object detection and tracking module is used to provide semantic information about the surroundings of the autonomous vehicle. This module consists of three parts: object detection, object clustering, and object tracking. We use Nvidia's proprietary DNN called *DriveNet* [21] to perform object detection. The input to the object detection network is RCCB (Red-Clear-Clear-Blue) image and the output is object proposals with bounding boxes. Each object can have multiple proposals; the object clustering algorithm clusters these multiple proposals into one bounding box for each detected object. The object tracking algorithm tracks the detected bounding boxes to maintain temporal consistency. It detects and tracks various six different classes of objects such as car, truck, person, bicycle, traffic sign, and road sign. It overlays bounding boxes on the detected objects. The color of the bounding boxes represents the classes that it detects are as follows: red for cars, cyan for trucks, green for persons, blue for bicycles, yellow for traffic signs, and magenta for road signs.

B. *Lane Detection*

A robust and accurate lane detection system is crucial to ADAS systems like Lane Keep Assist System (LKAS), Lane Departure Warning System (LDWS), and also provides vital information to autonomous vehicles. We use Nvidia's proprietary DNN called *LaneNet* [21] to perform lane detection. The input image format to this network is RCCB (Red-Clear-Clear-Blue) image and the output is polylines representing lane markings. It calculates a probability map of lane markings for each pixel using an encoder-decoder architecture on the input image. The map is then binarized into clusters of lane-markings, through which polylines are fitted to assign lane position types. It recognizes the four different types of lane markings, such as left adjacent-lane, left ego-lane, right ego-lane, and right adjacent-lane, when they are present on the road. The lane detection module overlays polylines on the detected lane markings. The colors of the polylines represent the lane marking types are as follows: yellow for left adjacent-lane, red for left ego-lane, green for right ego-lane, and blue for the right adjacent-lane.

C. *Free Space Detection*

Free space detection provides critical information about the drivable space to the navigation system of an autonomous vehicle. We use Nvidia's proprietary DNN called *OpenRoadNet* [21] to perform the free space detection. The input to the network is RCCB (Red-Clear-Clear-Blue) image and the output is a boundary across the image from left to right. The boundary separates the obstacle from open road space. Each pixel on the boundary is associated with one of the four semantic labels: red for vehicle, blue for pedestrian, green for curb, and yellow other.

V. Proposed Stereo Perception Framework

In this Section, we present the stereo-vision based perception framework for autonomous vehicles. The functional architecture of the stereo perception framework that is deployed in Nvidia Drive PX 2 platform is shown in Figure 2.

The input to the framework is a synchronized raw stereo image pair from a stereo camera or a video file. We use a custom made stereo camera manufactured using two AR0231 GMSL cameras. The stereo camera is calibrated using a stereo calibration tool, as mentioned in Section II, and the camera calibration parameters are read from the rig configuration file during the initialization of the framework. Camera synchronization is guaranteed as the ports on which the two cameras are connected, are hardware synchronized.

The images from the cameras are in Bayer RCCB (Red-Clear-Clear-Blue) format, which is converted to RGBA (Red-Green-Blue-Alpha) format before the rectification process. The left and right images are then undistorted and rectified, as explained the Section III-A. We use the stereo rectification functionality provided in the Nvidia DriveWorks software development kit (SDK) to perform the task. The rectified left and right camera images are converted to gray-scale images, and a pyramid of Gaussian images is built up to a specified

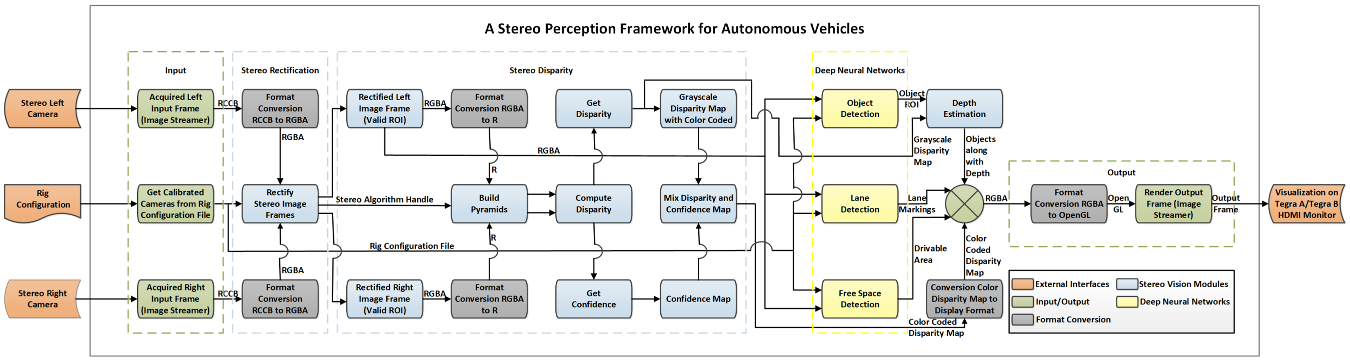

A Stereo Perception Framework for Autonomous Vehicles
Input
Stereo Rectification
Stereo Disparity
Deep Neural Networks
Output
Stereo Left Camera
Rig Configuration
Stereo Right Camera
Acquired Left Input Frame (Image Streamer)
Get Calibrated Cameras from Rig Configuration File
Acquired Right Input Frame (Image Streamer)
RCCB
Format Conversion RCCB to RGBA
RGBA
Rectify Stereo Image Frames
Rectified Left Image Frame (Valid ROI)
Rectified Right Image Frame (Valid ROI)
Format Conversion RGBA to R
R
Stereo Algorithm Handle
Build Pyramids
Rig Configuration File
Get Disparity
Compute Disparity
Get Confidence
Grayscale Disparity Map with Color Coded
Mix Disparity and Confidence Map
Confidence Map
Object Detection
Object ROI
Depth Estimation
Grayscale Disparity Map
Objects along with Depth
Lane Detection
Lane Markings
Free Space Detection
Drivable Area
Color Coded Disparity Map
Conversion Color Disparity Map to Display Format
Color Coded Disparity Map
Format Conversion RGBA to OpenGL
Open GL
Render Output Frame (Image Streamer)
Output Frame
Visualization on Tegra A/Tegra B HDMI Monitor
External Interfaces
Stereo Vision Modules
Input/Output
Deep Neural Networks
Format Conversion

Fig. 2: Functional Architecture of the Proposed Stereo Perception Framework.

level. The level 0 image of the pyramid or the full resolution gray-scale image is used for disparity computation. We use SSD pixel matching technique to find the stereo correspondence of every pixel of the left image with the right image to compute the disparity map with respect to the left image as explained in Section III-B. The Nvidia DriveWorks SDK provides the disparity computation library, and it returns both the disparity map and disparity confidence map of the left image. The disparity and confidence map are used to generate a colored disparity map, which is displayed as output, where the invalid pixels are displayed in black color.

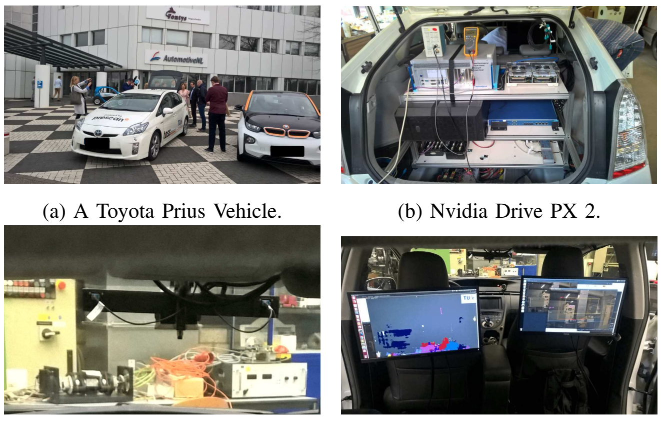
The rectified left camera image from the stereo rectifier is passed as input to the DNN module, as explained in Section IV. The object detector and tracker described in Section IV-A, outputs region of interest of detected objects as bounding boxes with its class. The depth estimator computes the depth of each detected object by utilizing the computed disparity map of the left image. We compute the disparity of the each object by computing average disparity of $1/3\ rd$ area of the bounding box around its centre. This filters out outlier near the edges of the bounding box. The depth of each object is then computed, as explained in Section III-C. The lane detector described in Section IV-B can classify four different lane markings within an image, and overlays the recognized lane markings on the output image. The free space detector described in Section IV-C identifies the drivable collision-free space within the image and overlays the identified drivable area with a separation boundary on the output image.

The output image is converted to an OpenGL image, and it overlays the output of the object detector and tracking, depth estimator, lane detector, and free-space detector on it before the image rendering process. The image rendering process renders the results from the previous modules in a meaningful way to the user through the in-vehicle Tegra A/Tegra B HDMI computer monitors.

VI. Experimental Setup

The TU/e–TASS International highly automated driving research prototype vehicle, based on a 3rd generation hybrid Toyota Prius, is used to deploy and demonstrate the proposed stereo perception framework.

This vehicle is equipped with a GMSL (Gigabit Multimedia Serial Link) stereo camera, a Nvidia Drive PX 2 hardware platform, Ubuntu 16.04 based computer with Intel Core-i7 7700k with Nvidia Titan XP and 32GB RAM, two LG 21 inch 60Hz 1920x1080 pixels Full HD IPS LCD HDMI computer monitors along with HDMI cable to connect with Drive PX 2, Logitech K400 Plus Wireless Touch Keyboard, 10 Gigabit Ethernet switch, Huawei 3G/4G/Wifi modem/router to provide internet to the computer and Drive PX 2, and CAT7 cable supports 10 Gigabit Ethernet protocol to connect the computer and Drive PX 2 via Ethernet switch. All of the equipment is powered using a 2000W 12V DC to 220V AC converter connected to the battery of the vehicle. Figure 3 shows the prototype vehicle along with the used hardware components.



(a) A Toyota Prius Vehicle.

(b) Nvidia Drive PX 2.

(c) GMSL Stereo Camera.

(d) HDMI Computer Monitors.

Fig. 3: An Autonomous Research Vehicle Platform: (a) A Toyota Prius Vehicle equipped with (b) Nvidia Drive PX 2 (in the trunk of a vehicle), (c) GMSL Stereo Camera (at the rear view mirror), and (d) HDMI Computer Monitors (at the back side of vehicle front seats).

We use a custom-built stereo camera, which is composed of two identical Sekonix GMSL Automotive Cameras SF3323 with an ONSEMI CMOS AR0231 image sensor [22], 1928x1208 resolution (2.3 Mega Pixel), 60 FOV (field of view), focal length 5.8 mm, baseline of 30 cm, with FAKRA (Fachkreis Automobil, a German Standard) connector. The stereo camera is firmly fixed using a rigid mounting bar, high up at the rear-view mirror position, at the inner center of the windshield, align the camera center vertically with the horizon.

We use the Nvidia Drive PX 2 AutoChauffeur as an embedded hardware platform, which contains the two parker SoC (System on Chip), called Tegra A and Tegra B, two discrete GPUs (dGPUs), two integrated GPUs (iGPUs), and Aurix TC297. The Drive PX 2 hardware platform is mounted in the trunk of a car.

The proposed stereo perception software framework is developed in C++ on an Ubuntu 16.04 LTS and deployed on a Drive PX 2 hardware platform with DriveWorks 0.6.67, CUDA 9.0, and CuDNN 7.3.0 library.

VII. Experimental Results

In this Section, we show the intermediate results of the proposed framework to get a better overview of how it works, is depicted in Figure 4.

The acquired left and right images from the stereo camera are displayed in Figure 4a and Figure 4b. The rectified left and right images from the stereo rectifier are displayed in Figure 4c and 4d. The computed confidence map with respect to the left disparity map from the stereo disparity is displayed in Figure 4e. The detected objects on the road along with the stereo depth are displayed in Figure 4f. The recognized lane markings on the road along with their classification are displayed in Figure 4g. The identified drivable free space on the road along with obstacles classification are displayed in Figure 4h. The detected objects along with depth, recognized lane markings, and identified free space simultaneously on the road by the proposed stereo perception framework, are shown in Figure 4i.

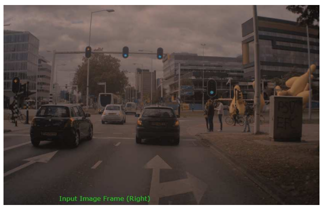
VIII. Performance Evaluation

In this Section, we evaluate the performance of the proposed stereo perception framework by analyzing the depth estimation output of detected objects and also its processing time on two different platforms.

A. *Depth Estimation*

We compare the depth output of the framework with the known distance of three different objects: a vehicle, a bicycle, and a person. The computed confidence map of the vehicle with respect to the left disparity map is displayed in Figure 5a, and the depth estimation of the vehicle is shown in Figure 5b. The computed confidence map of the bicycle with respect to the left disparity map is displayed in Figure 5c, and the depth estimation of the bicycle is shown in Figure 5d. The computed confidence map of the person with respect to the left disparity map is displayed in Figure 5e, and the depth estimation of the person is shown in Figure 5f. The depth estimation results along with actual depth, estimated depth, and depth error, are summarized in Table I.

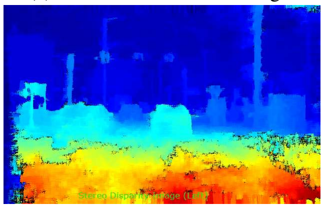
(a) Left stereo input image.



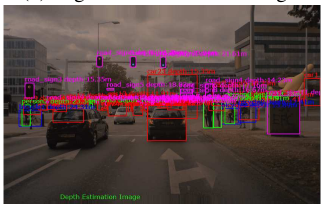
(b) Right stereo input image.

(c) Left stereo rectified image.

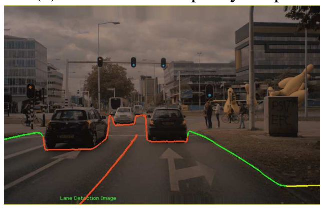
(d) Right stereo rectified image.



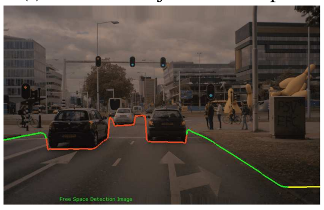
(e) Left stereo disparity map.



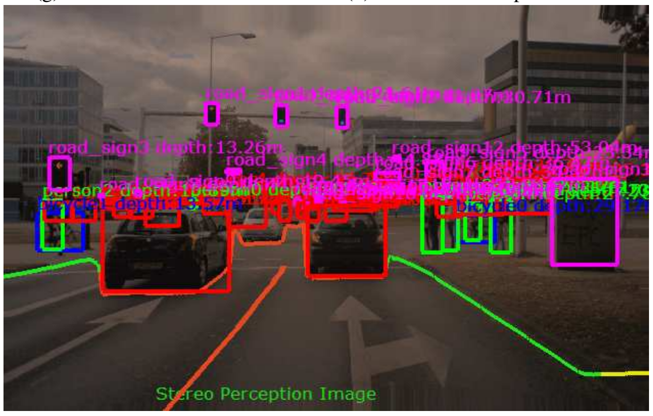
(f) Result of objects with depth.



(g) Result of lane detection.



(h) Result of free space detection.



(i) Result of proposed stereo perception framework.

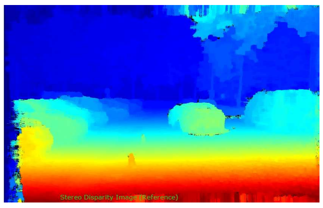
Fig. 4: Experimental Results: Proposed Stereo Perception Framework.

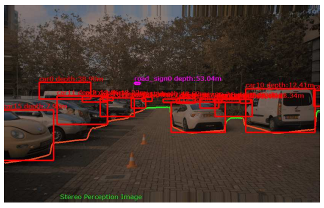
TABLE I: Depth Estimation Results (in meters).

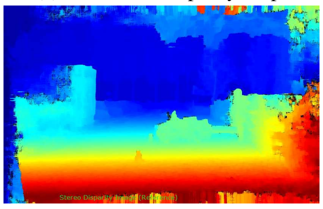
Objects	**Actual Depth**	**Estimated Depth**	**Depth Error**
Car	13.00	12.41	0.59
Bicycle	10.00	9.89	0.11
Person	7.00	6.71	0.29

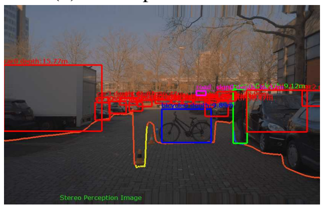
B. *Processing Time*

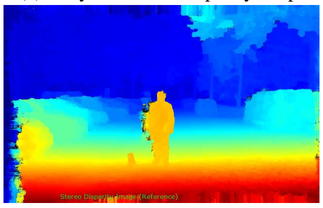
We compare the processing time of the proposed stereo perception framework with the Nvidia DNNs: DriveNet, LaneNet, and OpenRoadNet, on the Ubuntu 16.04 and Drive PX 2 platform, are shown in Table II. The processing time of the proposed framework is 99 ms (10.1 Hz) on a laptop with Quadro M1200 GPU and quad-core Intel Core i7 CPU


(a) Car stereo disparity map.


(b) Car depth estimation.


(c) Bicycle stereo disparity map.


(d) Bicycle depth estimation.


(e) Person stereo disparity map.

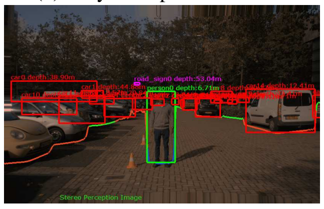

(f) Person depth estimation.

Fig. 5: Experimental Results: Proposed Depth Estimation.

running Ubuntu 16.04 (x86_64 architecture) and 134 ms (7.4 Hz) on the Drive PX 2 platform (aarch64 architecture), which is suitable for various low-speed ADAS applications.

TABLE II: Performance of frameworks (in milliseconds).

Platform (architecture)	**Nvidia DriveNet**	**Nvidia LaneNet**	**Nvidia OpenRoadNet**	**Stereo Perception**
Ubuntu16.04 (x86_64)	38	09	07	99
Drive PX2 (aarch64)	34	06	04	134

IX. Conclusions

In this paper, we proposed and developed a stereo perception for autonomous vehicles that runs real-time on the Nvidia Drive PX2 platform. We use images from a custom made stereo camera manufactured using two AR0231 GMSL cameras as input to the framework. The framework processes the stereo image pair to detect objects and estimate its depth, recognize lane boundaries, and identify drivable space simultaneously. It is deployed and tested on Drive PX2 platform in our prototype research vehicle to demonstrate the practical feasibility in real-time environment. The framework runs at 7.4 Hz on Drive PX 2 platform, which is suitable for various low-speed ADAS applications.

Acknowledgment

This research work is part of the i-CAVE (integrated cooperative automated vehicles) research programme within the Sensing, Mapping and Localization project (project number 363265/10024085). This i-CAVE programme is funded by NWO (Netherlands Organisation for Scientific Research).

References

[1] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: common practices and emerging technologies," *arXiv preprint arXiv:1906.05113*, 2019.

[2] G. P. Stein, O. Mano, and A. Shashua, "Vision-based acc with a single camera: bounds on range and range rate accuracy," in *IEEE IV2003 Intelligent Vehicles Symposium. Proceedings (Cat. No.03TH8683)*, June 2003, pp. 120–125.

[3] E. Dagan, O. Mano, G. P. Stein, and A. Shashua, "Forward collision warning with a single camera," in *IEEE Intelligent Vehicles Symposium, 2004*, June 2004, pp. 37–42.

[4] M. Haloi and D. B. Jayagopi, "A robust lane detection and departure warning system," in *2015 IEEE Intelligent Vehicles Symposium (IV)*, June 2015, pp. 126–131.

[5] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov, "Scalable object detection using deep neural networks," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[6] P. Tzirakis, G. Trigeorgis, M. A. Nicolaou, B. W. Schuller, and S. Zafeiriou, "End-to-end multimodal emotion recognition using deep neural networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 8, pp. 1301–1309, Dec 2017.

[7] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[8] C. Godard, O. Mac Aodha, and G. J. Brostow, "Unsupervised monocular depth estimation with left-right consistency," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[9] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

[10] H. Zhuang, R. Sudhakar, and J. yu Shieh, "Depth estimation from a sequence of monocular images with known camera motion," *Robotics and Autonomous Systems*, vol. 13, no. 2, pp. 87 – 95, 1994. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0921889094900515

[11] Nvidia Drive - Autonomous Vehicle Development Platforms, https://developer.nvidia.com/drive/, [Online], 2019.

[12] Z. Zhang, "Camera calibration," *Computer vision: a reference guide*, pp. 76–77, 2014.

[13] O. Faugeras, O. FAUGERAS, and M. I. of Technology, *Three-dimensional Computer Vision: A Geometric Viewpoint*, ser. Artificial intelligence. MIT Press, 1993. [Online]. Available: https://books.google.nl/books?id=Aa6TTW9dWy0C

[14] J.-Y. Bouguet, "Camera calibration toolbox for matlab (2008)," *URL http://www. vision. caltech. edu/bouguetj/calib_doc*, vol. 1080, 2008.

[15] P. Sturm, *Pinhole Camera Model*. Boston, MA: Springer US, 2014, pp. 610–613. [Online]. Available: https://doi.org/10.1007/978-0-387-31439-6_472

[16] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.", 2008.

[17] U. R. Dhond and J. K. Aggarwal, "Structure from stereo-a review," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 19, no. 6, pp. 1489–1510, Nov 1989.

[18] G. Xu and Z. Zhang, *Epipolar Geometry in Stereo, Motion, and Object Recognition: A Unified Approach*. USA: Kluwer Academic Publishers, 1996.

[19] H. Hirschmuller and D. Scharstein, "Evaluation of cost functions for stereo matching," in *2007 IEEE Conference on Computer Vision and Pattern Recognition*, June 2007, pp. 1–8.

[20] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[21] NVIDIA DriveWorks Development Guide, https://developer.nvidia.com/driveworks-docs/, [Online], 2019.

[22] Sekonix Camera Datasheets, https://developer.nvidia.com/driveworks/files/Sekonix_AR0231_2MP_SF332X_Automotive_GMSL_Camera_Datasheet_v2.2E.pdf, [Online], 2019.