Proc. of the International Conference on Electrical, Computer and Energy Technologies (ICECET)
9-10 December 2021, Cape Town-South Africa

A SysML-based Design and Development of Stereo Vision System with Pose and Velocity Estimation for Cooperative Automated Vehicles

Narsimlu Kemsaram
Department of Space Robotics
Space Robotics (SpaceR) Research Group
SnT Centre
University of Luxembourg
L-1855 Kirchberg, Luxembourg
Email: narsimlu.kemsaram@uni.lu
ORCID: 0000-0003-4672-6272

Anweshan Das
Department of Electrical Engineering
Mobile Perception Systems (MPS) Lab
Signal Processing Systems Group
Eindhoven University of Technology
5612 AZ Eindhoven, Netherlands
Email: anweshan.das@tue.nl
ORCID: 0000-0003-4008-7123

Gijs Dubbelman
Department of Electrical Engineering
Mobile Perception Systems (MPS) Lab
Signal Processing Systems Group
Eindhoven University of Technology
5612 AZ Eindhoven, Netherlands
Email: g.dubbelman@tue.nl
ORCID: 0000-0001-6635-3245

***Abstract*—Cooperative automated vehicles must perceive the environment accurately and have precise information about the leading vehicle's pose and velocity. This paper presents a SysML-based approach to design and develop a stereo vision-based perception system in an urban platooning scenario. It detects objects, lane markers, and free space in front of the follower vehicle using deep neural networks and computes the lead vehicle's relative pose and velocity. The relative pose is estimated using a geometric model-based pose estimation algorithm. The relative velocity is estimated from the change in pose within a known time. The lead vehicle's relative pose and velocity control the follower vehicle to follow the lead vehicle autonomously. Also, it displays the lead vehicle's pose and velocity information in a meaningful way on the in-vehicle display of the follower vehicle. The proposed system uses a custom-built automotive-grade stereo camera as input and runs on an automotive-grade embedded platform. It is tested on a simulation and prototype cooperative automated vehicle research platform. The evaluation results demonstrate that the proposed system operates in real-time and is suitable for cooperative automated vehicles.**

***Index Terms*—Autonomous vehicles, cooperative automated vehicles, deep neural networks, pose estimation, stereo vision system, systems modeling language, velocity estimation.**

I. Introduction

Researchers have made significant progress in Autonomous Vehicles (AVs) and Cooperative Automated Vehicles (CAVs) to reduce road accidents caused by human error and traffic congestion in recent years [1]. Especially, CAVs that interact with leader-follower vehicles attract significant attention in intelligent vehicle applications. An example of such cooperative behavior is platooning, where a group of vehicles driving autonomously in formation with fixed spacing. In a platooning scenario, the front vehicle is driven by the driver while the other vehicles follow the lead vehicle autonomously. Platooning can dramatically reduce the drag coefficient and fuel costs by reducing the spacing between the vehicles. Other benefits of platooning are the increase in road capacity and traffic safety [2]. The integrated Cooperative Automated Vehicles (i-CAVE) research project team has set up a Renault Twizy vehicle platooning, consisting of one lead vehicle and one follower vehicle, for this challenging task [3], is shown in Figure 1. The knowledge about the leading vehicle's pose and velocity is a vital prerequisite for developing platooning vehicles. In literature, several approaches use sensors like Camera, LiDAR, and Radar to solve these problems. In most approaches, platooning rely on a special infrastructure. For example, these approaches work well on highways. In the urban scenario,

Fig. 1: i-CAVE: A Renault Twizy Vehicle Platooning.

This research work was supported in part of the integrated Cooperative Automated Vehicles (i-CAVE) research programme within the sensing, mapping and localization project by the Netherlands Organisation for Scientific Research (NWO) under the Grant Number: 10024085.

these approaches can have many problems. In order to solve these problems, we use a custom-built stereo camera with a high dynamic range of capable automotive-grade cameras, a powerful automotive-grade computation platform, Artificial Intelligence (AI) based Deep Neural Networks (DNNs), and a geometric model-based pose and velocity estimation algorithm in the i-CAVE research project. The various Electronic Control Units (ECUs) are deployed in the i-CAVE research project to enforce digital control of functional aspects such as Stereo Camera, Radar, GPS, IMU, and Vehicle Control System. Since the number of ECUs grows in the i-CAVE research project, there is an increase in the complexity of managing the software. In order to manage the software complexity, we propose a Systems Modeling Language (SysML) based design methodology and model-driven algorithms.

The main objective of this paper is to design, develop, and evaluate an onboard stereo vision system using an automotive-grade stereo camera for cooperative automated vehicles.

The key contributions of this paper are:

- We design an onboard stereo vision system model with a SysML-based design methodology using the IBM Rational Rhapsody tool.
- The model-driven methods are derived from the designed model, developed, and integrated into an independent onboard stereo vision system. It can estimate the lead vehicle's pose and velocity.
- The performance of the onboard stereo vision system has been evaluated with the Carla simulator and cooperative automated research vehicles.

This paper is structured as follows. Section II provides an overview of the related works in the field of design methodologies and development algorithms. Section III describes the proposed onboard stereo vision system's primary functions in the form of a stakeholder's needs, functional and non-functional requirements. Section IV explains the SysML-based design methodology of the proposed system. Section V describes the model-driven development algorithms of the proposed system. Section VI presents the experimental results of the proposed system. Finally, Section VII concludes the paper.

II. Related Work

This section discusses the related work in design methodologies and developing algorithms for autonomous vehicles in detail.

A. Design Methodologies

In 1999, the Unified Modeling Language (UML) was introduced for the purpose of system design [4]. Later, UML was used in the development of a complex System-on-a-Chip (SoC) [5]. It was also utilized for platform-based design concepts in embedded systems [6]. Additionally, an approach is proposed for implementing embedded systems that unify UML and SoC design methodologies [7]. Later, a generic object-oriented framework has been proposed for real-time systems modeling based on the UML Real-Time (UML-RT) [8]. The Systems Modeling Language (SysML) is an extension of the UML used for system design modeling [9]. It is also more and more adopted by complex embedded systems such as mechatronics embedded systems [10], automotive embedded systems [11], and aerospace embedded systems [12]. In this paper, we use *a SysML-based design methodology* for stereo vision system design.

B. Development Algorithms

Vision-based pose estimation algorithms, such as non-model-based and model-based pose estimation algorithms [13], have been used to solve vehicle platooning problems in various ways. Non-model-based pose estimation algorithms include structure from motion [14], optic flow [15], and stereo ego-motion estimation [16]. Model-based pose estimation algorithms include feature-based model tracking [17], contour tracking [18], template matching [19], and Pose from Orthography and Scaling (POS) algorithm [20]. The POS method is fast, simple, and robust, even when camera calibration issues are present. It has the advantage of not necessitating a starting pose [21]. However, there is a limitation that does not work if the objects are flat or plane. To overcome this limitation, there is an advanced algorithm, which is coplanar POSIT (POS with Iterations) [22]. However, this is not possible with POSIT if the object correspondences are unknown. To overcome this problem, there is a new formulation of the POSIT algorithm, which is SoftPOSIT [23]. In this paper, we use the *SoftPOSIT development algorithm* for pose estimation.

III. Proposed Onboard Stereo Vision System

This section defines the proposed onboard stereo vision system's main functions using a Stakeholder's Needs Document (SND) that includes stakeholders' needs, functional and non-functional requirements.

A. Stakeholder's Needs

The proposed system is an ECU of the i-CAVE research project, consisting of an onboard AI-based acquisition and processing system. The system is intended to provide the environment perception within cooperative automated vehicles. The system provides consistent assistance to a follower vehicle during the vehicles' platooning. The purpose of the system is to detect the objects, identify lane markings, recognize drivable free space, estimate the lead vehicle's pose and velocity in front of the vehicle. The system also provides environment obstacle information for the vehicle control system and visualizes the obstacle information on the in-vehicle display. The following functional and non-functional requirements are met to meet the stakeholder's needs, as mentioned above.

B. Functional Requirements

The most important functional requirements of the proposed system are: i) system operational: on power-up, the system shall perform the functional activation and deactivation using the human-machine interface, ii) environmental perception: system shall perform the environmental perception functions

such as object detection, lane detection, and free space detection on activation, and iii) platooning operation: system shall estimate the lead vehicle's pose and velocity during the vehicle platooning operation. The system shall monitor the environment from a stereo camera and process it to send the obstacle information to the vehicle control system. The system shall visualize the obstacle information on the in-vehicle display.

C. Non-functional Requirements

The most important non-functional requirements of the proposed system are: i) functional: system must be capable of functioning in any situation and on any road surface, ii) usability: system must straightforward to use by any cooperative automated vehicle, iii) reliability: system must consistently perform the specified functions without failure, iv) performance: system must run on a low power consumption device and capable of executing with real-time performance, and v) supportability: system must have a modular design, a standardized interface, and easy to install in a wide range of vehicle platooning applications.

IV. SysML-based Design Methodology

In this section, a SysML-based design methodology [24] is presented. SysML models and diagrams are created by using the IBM Rational Rhapsody 8.3.1 tool.

The proposed design methodology aims to assist designers in creating consistent modeling in system design. It is a two-level design view modeling process: i) a black-box design view, which gives an external point of view of the proposed system, ii) a white-box design view, which gives an internal point of view of the proposed system. Each design view is made up of several activities. In each activity, a SysML diagram describes a specific point of view of the proposed system. More detail about each activity in each design view and how they are represented with SysML diagrams is explained below.

A. Black-Box Design

In this black-box design view, the proposed system is considered a black box. In this view, an external point of view design is performed to identify the system's requirements and specifications.

The key activities of this black-box design view are:

i) Identify Stakeholders Needs: This activity identifies the system requirements of the proposed system. It is usually in the form of an SND with the Stakeholder's needs, as mentioned in Section III-A.

ii) Elicit Requirements: This activity captures the system's requirements based on identified Stakeholder's needs. These system requirements are captured in the SysML Requirements Diagram (RD). The SysML RD contains functional and non-functional requirements. The functional requirements contain requirements for system operational, environmental perception, and platooning operation, as mentioned in Section III-B. The non-functional requirements contain functional, usability, reliability, performance, and supportability requirements, as mentioned in Section III-C. In this design view, we consider only functional requirements and do not consider non-functional requirements.

iii) Define System Context: This activity defines the system's context or boundary, which may interact with external systems directly or indirectly based on elicited requirements. This system context represent is represented in a SysML Block Definition Diagram (BDD), including the proposed system and its interactions with external interface actors such as stakeholders and external systems, shown in Figure 2. The

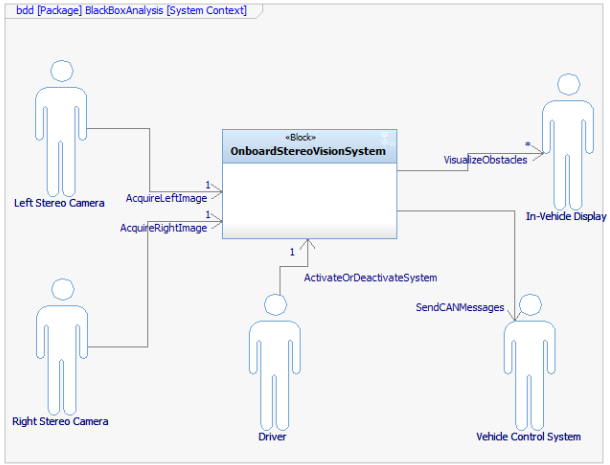

Fig. 2: Stereo Vision System Context Diagram.

Onboard Stereo Vision System is the system of interest in the BDD, and it provides the context for the proposed system. The other stakeholders are external systems to the proposed system. These include the *Driver*, *Left Stereo Camera*, *Right Stereo Camera*, *Vehicle Control System*, and *In-Vehicle Display*. Its relationships as defined by its associations.

iv) Define External Interfaces: This activity defines the system's external interfaces based on defined system context and interactions with external systems. The *Onboard Stereo Vision System* is the system of interest block, and the other systems are external interface actors to the system block. These include the *Driver*, *Left Stereo Camera*, *Right Stereo Camera*, *Vehicle Control System*, and *In-Vehicle Display*. Its interfaces as defined by its proxy ports.

v) Identify Use Cases and Actors: This activity defines the proposed system's use cases and actors based on the defined system's external interfaces. The association relationship between actors and use cases in a Use Case Diagram (UCD), shown in Figure 3. In the UCD, the *Perceive Environment* use case represents the system's main functionality that is always performed when the *Driver* performs the *Activate System* use case. The *Activate System*, *Stereo Vision*, *Detect and Classify Objects*, *Estimate Pose*, *Estimate Velocity*, *Identify Lane Markings*, *Recognize Free Space Boundary*, *Send Obstacle Information*, and *Deactivate System* use cases include the *Perceive Environment* use case. The *Visualize Obstacle Information* use case extends the *Perceive Environment* use

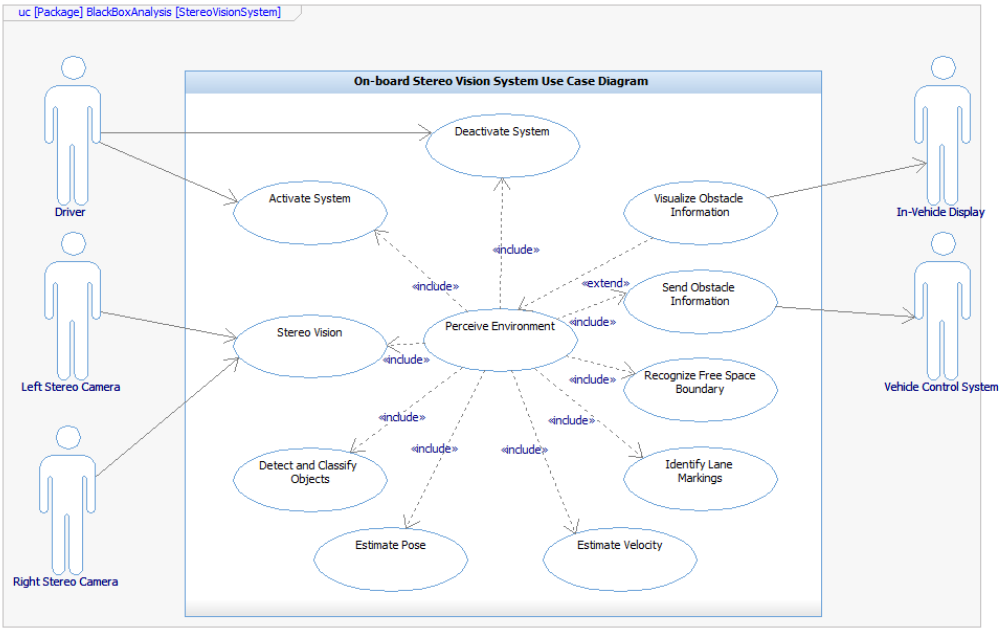

Fig. 3: Stereo Vision System Use Case Diagram.

case. The active actor is *Driver*, and the passive actors are the *Left Stereo Camera*, *Right Stereo Camera*, *Vehicle Control System*, and *In-Vehicle Display*.

vi) Define Functional Scenarios: This activity defines the proposed system's sequential description of a functional scenario with a Sequence Diagram (SD) for each identified use case. In the SD, the interactions between the system and its external interfaces are detailed. The behavior for the *Perceive Environment* use case from the UCD in Figure 3 is represented in the SD in Figure 4. The SD shows the *Driver* sending an *Activate System* message requesting the *Onboard Stereo Vision System* to activate. The *Onboard Stereo Vision System* responds with the *System Activated* reply message shown as a dashed line. This is followed by the *Left Stereo Camera*, *Right Stereo Camera*, and *Onboard Stereo Vision System* interactions to acquire images. The three interactions, *Object Perception*, *Lane Perception*, *Free Space Perception*, occur parallel to perceive the obstacles. The *Onboard Stereo Vision System* estimates the detected object's pose and velocity estimation using *EstimatePose* and *EstimateVelocity*. Also, it sends this information to the *Vehicle Control System* and displays it on the *In-Vehicle Display*. The last interaction is *Deactivate System* by the *Driver*. The SD shows the Driver sending a *Deactivate System* message requesting the *Onboard Stereo Vision System* to deactivate. The *Onboard Stereo Vision System* responds with the *System Deactivated* reply message shown as a dashed line.

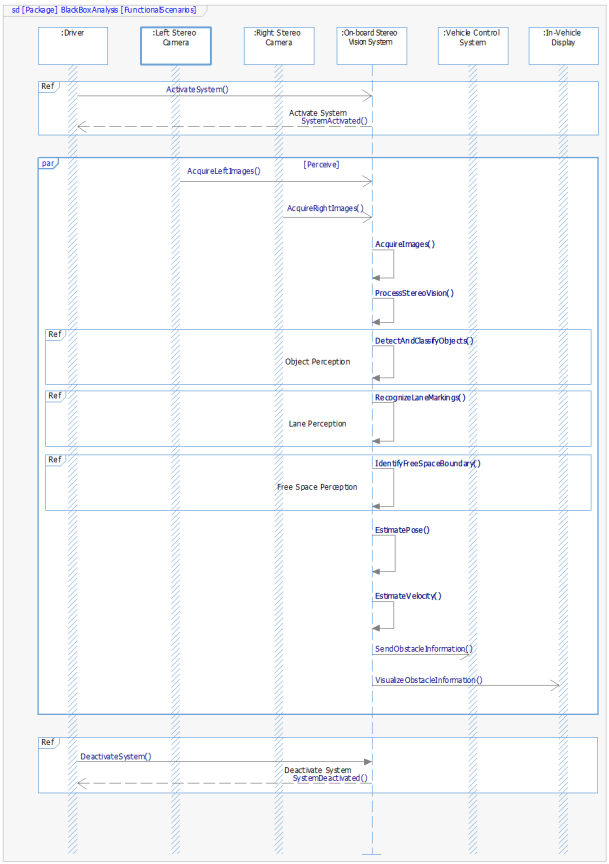

Fig. 4: Stereo Vision System Sequence Diagram.

vii) Identify System States and Modes: This activity defines the proposed system's operating states and modes based on the defined system's behavior in the SD. The proposed system's State Machine Diagram (STM) is shown in Figure 5. In the STM, the system operates in three modes: *Activate*, *Perceive*, and *Deactivate*, and six states: *Initialize*, *Process*, *Update*,

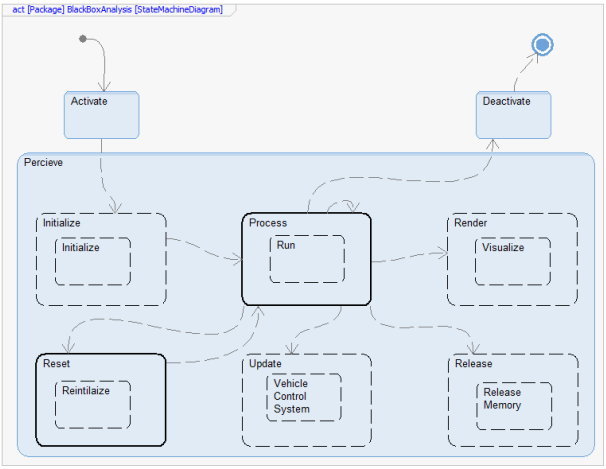

Fig. 5: Stereo Vision System State Machine Diagram. Modes: *Activate*, *Perceive*, *Deactivate*, and States: *Initialize*, *Process*, *Update*, *Render*, *Reset*, *Release*.

Render, *Reset*, and *Release*.

viii) Analyze Requirements with Use Cases: This activity analyzes the proposed system's user-level requirements and system-level requirements and ensures the traceability links between them based on their relationships, such as *derive*, *refine*, and *satisfy*.

B. White-Box Design

In this white-box design view, an internal point of view design is performed to model the system's structure and behavior based on the black-box view's design.

The key activities of this white-box design view are:

i) Identify Internal Functions: This activity identifies the proposed system set of internal functions based on the identified system operating states and modes. The system's internal functions are represented through activities and linked to its operations. The functional architecture is represented with an Activity Diagram (ACT) in Figure 6, which describes internal system functions.

ii) Define Logical Components: This activity defines the proposed system set of logical components based on the identified system set of internal functions. The system's logical components are represented through classes and linked to specified functions in the functional architecture. The logical architecture is defined with an IBD in Figure 7, which describes internal system components and flows between components.

iii) Define Physical Components: This activity defines the proposed system's physical components based on the identified system's logical components. The system's physical components are represented through components and ports and linked to specified components in the logical architecture.

V. Model-Driven Development Algorithms

This section presents developed algorithms to solve the vehicle platooning using an onboard stereo vision system for the i-CAVE research project.

Figure 8 shows that a follower vehicle is equipped with an onboard stereo camera and following a lead vehicle. The pose estimation method presented in this paper is used to estimate the lead vehicle's relative pose and velocity with respect to the follower vehicle by tracking the feature points of the lead vehicle. The relative pose and velocity estimation allow the follower to maintain a safe distance between the lead and follower vehicle. Based on the relative pose and velocity estimation, the throttle and brakes can be controlled with an actuation command of the follower vehicle control system to avoid collision with the lead vehicle or losing track of the lead vehicle. Also, it displays these results to the driver on the in-vehicle display. The below algorithms are implemented in C++ on an Ubuntu 16.04 LTS and deployed on an Nvidia Drive PX2 automotive embedded hardware platform with OpenCV 3.4.9, DrvieWorks 1.2, CUDA 9.2, cuDNN 7.4.1, Armadillo, and Boost libraries.

A. Object Perception

The object perception provides the obstacle information to the navigation system in front of the follower vehicle. We use Nvidia's deep learning-based DriveNet DNN to perform real-time object perception. Through the use of a DNN, the detection and classification of objects are computed efficiently and quickly. The input to this algorithm is a Red-Clear-Blue (RCB) image, and the output is an original image plus a list of detected objects with the following attributes: bounding box, object class, and confidence of detection. It detects and classifies objects such as cars, bicycles, pedestrians, road signs, and traffic lights. The bounding boxes' color represents the classes it detects: i) red for cars, ii) blue for bicycles, iii) orange for pedestrians, iv) green for road signs and v) magenta for traffic lights.

B. Pose Estimation

A geometric model for relative pose and velocity estimation using a stereo camera is shown in Figure 9. The relative pose is estimated using the SoftPOSIT pose estimation algorithm from a single image [23]. This algorithm requires at least four known pairs of 3D feature point coordinates and their corresponding 2D image coordinates and the camera's focal length. This algorithm generates a linear equation based on known 3D feature point coordinate pairs using a perspective projection model. This algorithm is in an iteration loop, repeated until it reaches the tolerance with the least error to give out the optimum solution of the pose estimation in terms of camera coordinates. This algorithm takes the input image, list of object points, list of image points and returns the $3x3$ rotation matrix (R) and the $3x1$ translation vector (T).

Rotation matrix,

$$R = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \quad (1)$$

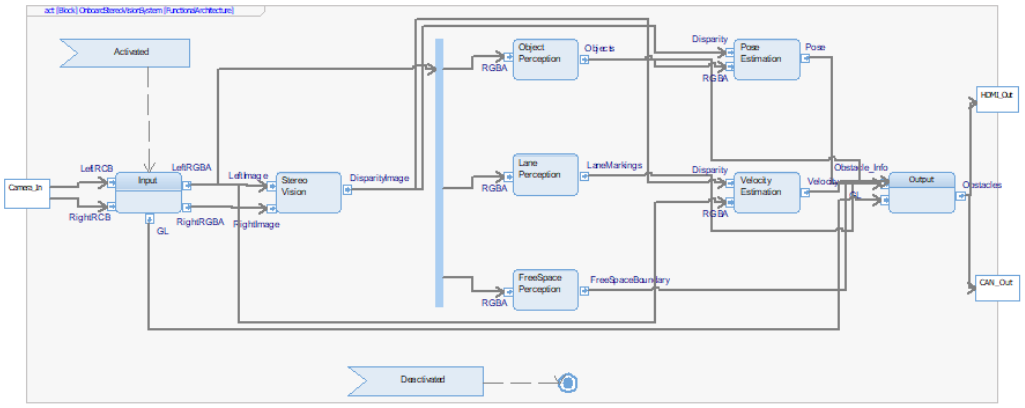

Fig. 6: Stereo Vision System Functional Architecture.

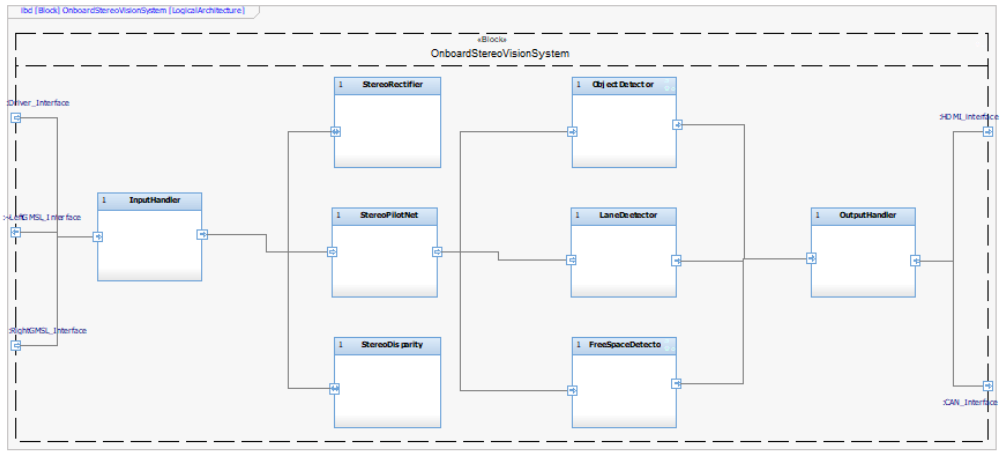

Fig. 7: Stereo Vision System Logical Architecture.

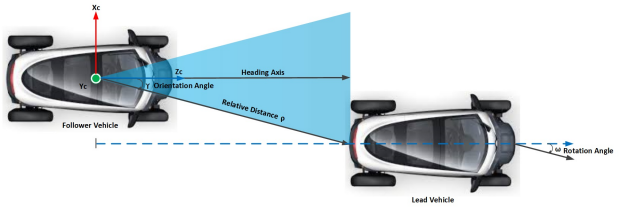

Fig. 8: A Geometric Model for Relative Pose and Velocity Estimation using a Stereo Camera in Vehicle Platooning (Pictorial View).

Translation vector,

$$T = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} \tag{2}$$

Rotation Angle: After obtaining the lead vehicle's rotation matrix (R) with respect to the follower vehicle in a vehicle platooning, the rotation angle is computed using the following steps. First, we can derive the Euler angles (θ_x, θ_y, θ_z) from the rotation matrx:

Pan,

$$\theta_x = atan2(R_{32}/R_{33}) \tag{3}$$

Tilt,

$$\theta_y = atan2(-R_{31}/\sqrt{R_{32}^2 + R_{33}^2}) \tag{4}$$

Roll,

$$\theta_z = atan2(R_{21}/R_{11}) \tag{5}$$

Euler angles are,

$$\theta = \begin{bmatrix} \theta_x & \theta_y & \theta_z \end{bmatrix} \tag{6}$$

where the elements of θ hold the rotation angles around the X, Y, and Z-axis, respectively. From the Euler angles, we can estimate the rotation angle,

$$\omega = \theta_z \tag{7}$$

Relative Distance: The translation vector, T represents the position of the object's reference point, P, with respect to the current origin, O. The length of this vector equals the distance between these two reference points (OP):

Relative distance,

$$\rho = \sqrt{T_x^2 + T_y^2 + T_z^2} \tag{8}$$

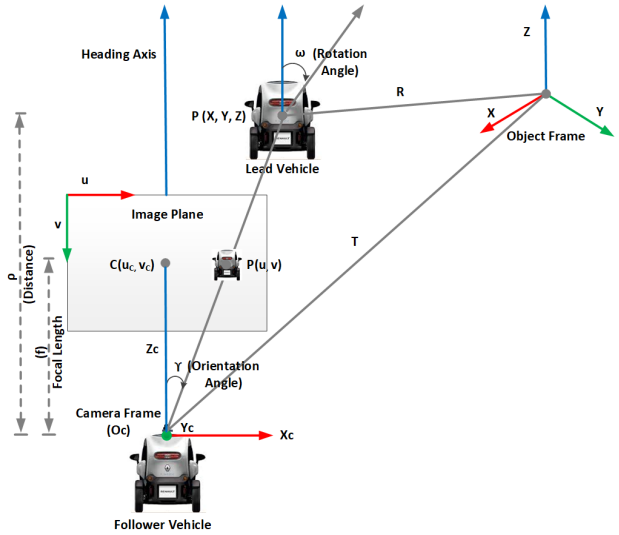

Fig. 9: A Geometric Model for Relative Pose and Velocity Estimation using a Stereo Camera in Vehicle Platooning (Front View). The reference camera coordinate system is right-handed, with the x-axis points to the right of the image plane, the y-axis points to the bottom of the image plane, and the z-axis points towards the center of the image plane.

Orientation Angle: The orientation angle can be calculated using the translation vector, T, in the XY-plane between the X-axis and T. The orientation is defined as the angle in the XY-plane between the X-axis and T.

Orientation angle,

$$\gamma = arctan(\triangle Y / \triangle X) \tag{9}$$

C. Velocity Estimation

The relative velocity (v) of the lead vehicle can be estimated using the relative distance and timestamps:

$$v = (d / \triangle t) \tag{10}$$

where d is the lead vehicle travelled distance within known time $\triangle$t, and v the relative velocity.

D. Lane Perception

The lane perception provides the lane markings in front of the follower vehicle to the navigation system. We use Nvidia's deep learning-based LaneNet DNN to perform real-time lane perception. Using a DNN, the detection and classification of lane markings are computed efficiently and quickly. The input to this algorithm is an RCB image, and the output is an original image plus a list of lane markings with a lane position and appearance type. It classifies a lane marking into one of the following five-position types: left adjacent lane, left ego-lane, right ego-lane, right adjacent lane, and undefined position. It detects a lane marking's appearance from the following four types: i) solid lane marking, ii) dashed lane marking, iii) road boundary, and iv) undefined appearance. It overlays the polylines on the lane markers detected. The polylines' colors represent the lane marking types: i) yellow for the left adjacent lane, ii) red for the left ego-lane, iii) green for the right ego-lane and iv) blue for the right adjacent lane.

E. Free Space Perception

The free space perception provides the drivable free space to the navigation system in front of the follower vehicle. We use Nvidia's deep learning-based FreeSpaceNet DNN to perform real-time free space perception. Using a DNN, the detection and classification of free space are computed efficiently and quickly. The input to this algorithm is an RCB image, and the output is an original image plus a free space boundary for the drivable free space. The recognized free space boundary divides the obstacles from open drivable free space. Each pixel on the boundary has a semantic label: i) green for the curb, ii) blue for pedestrians, iii) red for cars, and iv) yellow for others.

VI. Experimental Results

This section covers the proposed onboard stereo vision system's experimental results on simulation and prototype research vehicles.

A. Simulation

The proposed system acquires synthetic images from the virtual stereo camera in the Carla simulation environment. The simulation results of the proposed system are shown in Figure 10.

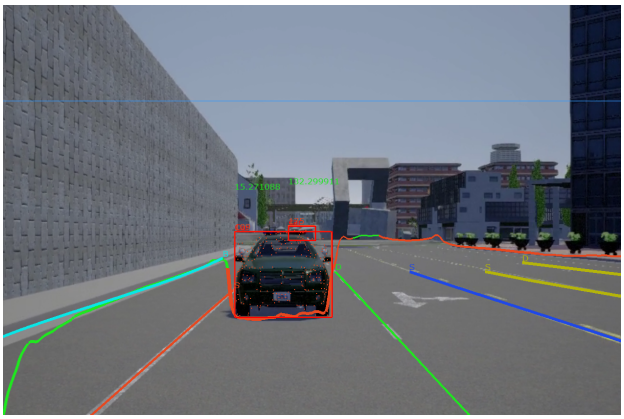

Fig. 10: Results of Stereo Vision System with Simulation.

B. Research Vehicle

The proposed system acquires input images from the on-board stereo camera in a real-world environment. The experimental results of the proposed system are shown in Figure 11.

C. Performance Evaluation

The processing time of the proposed system with Carla simulation and research vehicle is shown in Table I. The proposed system's processing time on the host computer with the Intel Core i7 CPU, the Nvidia TITAN Xp GPU card is 23

Fig. 11: Results of Stereo Vision System with Vehicle.

TABLE I: Processing Time (in milliseconds).

Platform	Perception Deep Neural Networks			Stereo Vision System
	Object Perception	Lane Perception	Free Space Perception	
Simulation	11	03	01	23
Vehicle	34	06	04	138

ms (43 Hz), and the target Drive PX2 computer is 138 ms (7.2 Hz), which is suitable for low-speed cooperative automated vehicles.

VII. Conclusion

This paper presented a SysML-based design and development of an onboard stereo system for cooperative automated vehicles. The SysML-based design simplifies the complexity of stereo vision software. The proposed design model fulfills the functional and non-function requirements of the Stakeholder's needs. The model-driven methods are derived from the designed model, developed, and integrated into an independent electronic control unit. The experimental evaluation has been performed on the Carla simulation and cooperative automated vehicle research platform. The experimental result shows that the proposed system classifies objects, identifies lane markings, recognizes drivable-free space boundary, and estimates the relative pose and velocity. The proposed system runs at 43 Hz (23 ms) on a high-end computer with a Titan Xp graphics card. It achieves more than 7.2 Hz (138 ms) on an Nvidia Drive PX2 automotive-grade hardware platform in real-time. However, an extension of the perception system with a combination of a Stereo camera and Radar to improve the estimation of pose and velocity is desirable. Therefore, future work should include a follow-up work designed to evaluate whether the sensor fusion of the Stereo camera and Radar operates in real-time on an Nvidia Drive PX2 automotive-grade embedded platform.

Acknowledgment

The authors would like to thank the entire integrated Cooperative Automated Vehicles (i-CAVE) research team, without whom the project would not have been possible.

References

[1] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE Access*, vol. 8, pp. 58 443–58 469, 2020.

[2] J. Ploeg and R. de Haan, "Cooperative automated driving: from platooning to maneuvering," in *5th International Conference on Vehicle Technology and Intelligent Transport Systems, VEHITS 2019*. SCITEPRESS-Science and Technology Publications, Lda., 2019, pp. 5–10.

[3] Integrated Cooperative Automated Vehicles (i-CAVE) project, https://i-cave.nl/, [Online], 2019.

[4] G. Martin, "Uml and vcc. cadence design systems," *Inc., White Paper*, 1999.

[5] T. Moore, Y. Vanderperren, G. Sonck, P. Van Oostende, M. Pauwels, and W. Dehaene, "A design methodology for the development of a complex system-on-chip using uml and executable system models," in *Proc. of ECSL'02*, 2002.

[6] R. Chen, M. Sgroi, L. Lavagno, G. Martin, A. Sangiovanni-Vincentelli, and J. Rabaey, "Uml for real: design of embedded real-time systems, chapter 5–uml and platform-based design," 2003.

[7] M. Edwards and P. Green, "Uml for hardware and software object modeling," in *UML for Real*. Springer, 2003, pp. 127–147.

[8] B. Selic, "A generic framework for modeling resources with uml," *Computer*, vol. 33, no. 6, pp. 64–69, 2000.

[9] F. Mhenni, "Safety analysis integration in a systems engineering approach for mechatronic systems design," Ph.D. dissertation, Ecole Centrale Paris, 2014.

[10] F. Mhenni, J.-Y. Choley, O. Penas, R. Plateaux, and M. Hammadi, "A sysml-based methodology for mechatronic systems architectural design," *Advanced Engineering Informatics*, vol. 28, no. 3, pp. 218–231, 2014.

[11] J. D'Ambrosio and G. Soremekun, "Systems engineering challenges and mbse opportunities for automotive system design," in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2017, pp. 2075–2080.

[12] X. Fei, C. Bin, L. Rui, and H. Shunhua, "A model-based system engineering approach for aviation system design by applying sysml modeling," in *2020 Chinese Control And Decision Conference (CCDC)*, 2020, pp. 1361–1366.

[13] J. Shi, S. Ulrich, and S. Ruel, "Spacecraft pose estimation using a monocular camera," in *67th International Astronautical Congress, Guadalajara, Mexico*, 2016.

[14] C. Tomasi and T. Kanade, "Shape and motion from image streams under orthography: a factorization method," *International journal of computer vision*, vol. 9, no. 2, pp. 137–154, 1992.

[15] D. Sinclair, A. Blake, and D. Murray, "Robust estimation of egomotion from normal flow," *International Journal of Computer Vision*, vol. 13, no. 1, pp. 57–69, 1994.

[16] L.-P. Morency and R. Gupta, "Robust real-time egomotion from stereo images," in *Proceedings 2003 International Conference on Image Processing (Cat. No. 03CH37429)*, vol. 2. IEEE, 2003, pp. II–719.

[17] D. B. Gennery, "Visual tracking of known three-dimensional objects," *International Journal of Computer Vision*, vol. 7, no. 3, pp. 243–270, 1992.

[18] M. Isard and A. Blake, "Condensation—conditional density propagation for visual tracking," *International journal of computer vision*, vol. 29, no. 1, pp. 5–28, 1998.

[19] F. Jurie, M. Dhome *et al.*, "Real time robust template matching." in *BMVC*, vol. 2002, 2002, pp. 123–132.

[20] D. F. Dementhon and L. S. Davis, "Model-based object pose in 25 lines of code," *International journal of computer vision*, vol. 15, no. 1-2, pp. 123–141, 1995.

[21] D. Grest, T. Petersen, and V. Krüger, "A comparison of iterative 2d-3d pose estimation methods for real-time applications," in *Scandinavian Conference on Image Analysis*. Springer, 2009, pp. 706–715.

[22] D. Oberkampf, D. F. DeMenthon, and L. S. Davis, "Iterative pose estimation using coplanar feature points," *Computer Vision and Image Understanding*, vol. 63, no. 3, pp. 495–511, 1996.

[23] P. David, D. Dementhon, R. Duraiswami, and H. Samet, "Softposit: Simultaneous pose and correspondence determination," *International Journal of Computer Vision*, vol. 59, no. 3, pp. 259–284, 2004.

[24] S. Friedenthal, A. Moore, and R. Steiner, *A practical guide to SysML: the systems modeling language*. Morgan Kaufmann, 2014.