DAFTAR PUSTAKA

A, C., M, D., 2022. Headlight Control Using Deep Learning-Based Vehicle Detection. Int. J. Res. Appl. Sci. Eng. Technol. 10, 1819–1823. https://doi.org/10.22214/ijraset.2022.48323

Abdullah, A., Oothariasamy, J., 2020. Vehicle Counting using Deep Learning Models: A Comparative Study. Int. J. Adv. Comput. Sci. Appl. 11. https://doi.org/10.14569/IJACSA.2020.0110784

Adz-Dzikri, A.A., Virgono, A., Dirgantara, F.M., 2021. Advance Driving Assistance Systems: Object Detection and Distance Estimation Using Deep Learning, in: 2021 8th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI). Presented at the 2021 8th International Conference on Electrical Engineering, Computer Science and Informatics (EECSI), IEEE, Semarang, Indonesia, pp. 381–386. https://doi.org/10.23919/EECSI53397.2021.9624218

Afshar, M.F., Shirmohammadi, Z., Ghahramani, S.A.A.G., Noorparvar, A., Hemmatyar, A.M.A., 2023. An Efficient Approach to Monocular Depth Estimation for Autonomous Vehicle Perception Systems. Sustainability 15, 8897. https://doi.org/10.3390/su15118897

Ajao, Q., Oludamilare, O., Prio, M.H., 2023. Safety Challenges and Analysis of Autonomous Electric Vehicle Development: Insights from On-Road Testing and Accident Reports. SSRN Electron. J. https://doi.org/10.2139/ssrn.4454486

Aleem, S., Kumar, T., Little, S., Bendechache, M., Brennan, R., McGuinness, K., 2022. Random Data Augmentation based Enhancement: A Generalized Enhancement Approach for Medical Datasets.

AliYev, N., Güzel, M.T., Sezer, O., 2022. Realization of the Autonomous Driving System on the Experimental Vehicle. Acad. Platf. J. Eng. Smart Syst. 10, 48–56. https://doi.org/10.21541/apjess.1060763

Alzubaidi, L., Zhang, J., Humaidi, A.J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M.A., Al-Amidie, M., Farhan, L., 2021. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. J. Big Data 8, 53. https://doi.org/10.1186/s40537-021-00444-8

Arabi, S., Sharma, A., Reyes, M., Hamann, C., Peek-Asa, C., 2022. Farm Vehicle Following Distance Estimation Using Deep Learning and Monocular Camera Images. Sensors 22, 2736. https://doi.org/10.3390/s22072736

Azurmendi, I., Zulueta, E., Lopez-Guede, J.M., González, M., 2023. Simultaneous Object Detection and Distance Estimation for Indoor Autonomous Vehicles. Electronics 12, 4719. https://doi.org/10.3390/electronics12234719

Bachute, M.R., Subhedar, J.M., 2021. Autonomous Driving Architectures: Insights of Machine Learning and Deep Learning Algorithms. Mach. Learn. Appl. 6, 100164. https://doi.org/10.1016/j.mlwa.2021.100164

Baczmanski, M., Synoczek, R., Wasala, M., Kryjak, T., 2023. Detection-segmentation convolutional neural network for autonomous vehicle perception, in: 2023 27th International Conference on Methods and Models

in Automation and Robotics (MMAR). pp. 117–122. https://doi.org/10.1109/MMAR58394.2023.10242398

Berwo, M.A., Khan, A., Fang, Y., Fahim, H., Javaid, S., Mahmood, J., Abideen, Z.U., M.S., S., 2023. Deep Learning Techniques for Vehicle Detection and Classification from Images/Videos: A Survey. Sensors 23, 4832. https://doi.org/10.3390/s23104832

Bochkovskiy, A., Wang, C.-Y., Liao, H.-Y.M., 2020. YOLOv4: Optimal Speed and Accuracy of Object Detection.

Bourja, O., Derrouz, H., Abdelali, H.A., Maach, A., Thami, R.O.H., Bourzeix, F., 2021. Real Time Vehicle Detection, Tracking, and Inter-vehicle Distance Estimation based on Stereovision and Deep Learning using YOLOv3. Int. J. Adv. Comput. Sci. Appl. 12. https://doi.org/10.14569/IJACSA.2021.01208101

Cepni, S., Atik, M.E., Duran, Z., 2020. Vehicle Detection Using Different Deep Learning Algorithms from Image Sequence. Balt. J. Mod. Comput. 8. https://doi.org/10.22364/bjmc.2020.8.2.10

Chen, K.-H., Shou, T.D., Li, J.K.-H., Tsai, C.-M., 2018. Vehicles Detection On Expressway Via Deep Learning: Single Shot Multibox Object Detector, in: 2018 International Conference on Machine Learning and Cybernetics (ICMLC). Presented at the 2018 International Conference on Machine Learning and Cybernetics (ICMLC), IEEE, Chengdu, pp. 467–473. https://doi.org/10.1109/ICMLC.2018.8526958

Chen, Y., Li, Z., 2022. An Effective Approach of Vehicle Detection Using Deep Learning. Comput. Intell. Neurosci. 2022, 1–9. https://doi.org/10.1155/2022/2019257

Chen, Z., Khemmar, R., Decoux, B., Atahouet, A., Ertaud, J.-Y., 2019. Real Time Object Detection, Tracking, and Distance and Motion Estimation based on Deep Learning: Application to Smart Mobility, in: 2019 Eighth International Conference on Emerging Security Technologies (EST). Presented at the 2019 Eighth International Conference on Emerging Security Technologies (EST), IEEE, Colchester, United Kingdom, pp. 1–6. https://doi.org/10.1109/EST.2019.8806222

Ciberlin, J., Grbic, R., Teslic, N., Pilipovic, M., 2019. Object detection and object tracking in front of the vehicle using front view camera, in: 2019 Zooming Innovation in Consumer Technologies Conference (ZINC). Presented at the 2019 Zooming Innovation in Consumer Technologies Conference (ZINC), IEEE, Novi Sad, Serbia, pp. 27–32. https://doi.org/10.1109/ZINC.2019.8769367

Ding, M., Zhang, Z., Jiang, X., Cao, Y., 2020. Vision-Based Distance Measurement in Advanced Driving Assistance Systems. Appl. Sci. 10, 7276. https://doi.org/10.3390/app10207276

Duran, O., Turan, B., 2022. Vehicle-to-vehicle distance estimation using artificial neural network and a toe-in-style stereo camera. Measurement 190, 110732. https://doi.org/10.1016/j.measurement.2022.110732

Elawady, I., Özcan, C., 2022. A new effective denoising filter for high density impulse noise reduction. Turk. J. Electr. Eng. Comput. Sci. 30, 1388–1403.

https://doi.org/10.55730/1300-0632.3855

Elzayat, I.M., Ahmed Saad, M., Mostafa, M.M., Mahmoud Hassan, R., El Munim, H.A., Ghoneima, M., Darweesh, M.S., Mostafa, H., 2018. Real-Time Car Detection-Based Depth Estimation Using Mono Camera, in: 2018 30th International Conference on Microelectronics (ICM). Presented at the 2018 30th International Conference on Microelectronics (ICM), IEEE, Sousse, Tunisia, pp. 248–251. https://doi.org/10.1109/ICM.2018.8704024

Espinosa, J.E., Velastin, S.A., Branch, J.W., 2017. Vehicle Detection Using Alex Net and Faster R-CNN Deep Learning Models: A Comparative Study, in: Badioze Zaman, H., Robinson, P., Smeaton, A.F., Shih, T.K., Velastin, S., Terutoshi, T., Jaafar, A., Mohamad Ali, N. (Eds.), Advances in Visual Informatics, Lecture Notes in Computer Science. Springer International Publishing, Cham, pp. 3–15. https://doi.org/10.1007/978-3-319-70010-6_1

Estrada, J., Opina Jr, G., Tripathi, A., 2021. Object and Traffic Light Recognition Model Development Using Multi-GPU Architecture for Autonomous Bus, in: Tallón-Ballesteros, A.J. (Ed.), Frontiers in Artificial Intelligence and Applications. IOS Press. https://doi.org/10.3233/FAIA210286

Farid, A., Hussain, F., Khan, K., Shahzad, M., Khan, U., Mahmood, Z., 2023. A Fast and Accurate Real-Time Vehicle Detection Method Using Deep Learning for Unconstrained Environments. Appl. Sci. 13, 3059. https://doi.org/10.3390/app13053059

Francés, L., Quintero, J., Fernández, A., Ruiz, A., Caules, J., Fillon, G., Hervás, A., Soler, C.V., 2022. Current state of knowledge on the prevalence of neurodevelopmental disorders in childhood according to the DSM-5: a systematic review in accordance with the PRISMA criteria. Child Adolesc. Psychiatry Ment. Health 16, 27. https://doi.org/10.1186/s13034-022-00462-1

Fregin, A., Muller, J., Dietmayer, K., 2017. Three ways of using stereo vision for traffic light recognition, in: 2017 IEEE Intelligent Vehicles Symposium (IV). Presented at the 2017 IEEE Intelligent Vehicles Symposium (IV), IEEE, Los Angeles, CA, USA, pp. 430–436. https://doi.org/10.1109/IVS.2017.7995756

George, M.A., George, A.M., 2014. Stereovision for 3D Information, in: Babu, B.V., Nagar, A., Deep, K., Pant, M., Bansal, J.C., Ray, K., Gupta, U. (Eds.), Proceedings of the Second International Conference on Soft Computing for Problem Solving (SocProS 2012), December 28-30, 2012, Advances in Intelligent Systems and Computing. Springer India, New Delhi, pp. 1595–1602. https://doi.org/10.1007/978-81-322-1602-5_158

Giovannoli, J., Martella, D., Federico, F., Pirchio, S., Casagrande, M., 2020. The Impact of Bilingualism on Executive Functions in Children and Adolescents: A Systematic Review Based on the PRISMA Method. Front. Psychol. 11, 574789. https://doi.org/10.3389/fpsyg.2020.574789

Guo, Q., Liu, J., Kaliuzhnyi, M., 2022. YOLOX-SAR: High-Precision Object Detection System Based on Visible and Infrared Sensors for SAR Remote Sensing. IEEE Sens. J. 22, 17243–17253. https://doi.org/10.1109/JSEN.2022.3186889

Gupta, A., Anpalagan, A., Guan, L., Khwaja, A.S., 2021. Deep learning for object detection and scene perception in self-driving cars: Survey, challenges, and open issues. Array 10, 100057. https://doi.org/10.1016/j.array.2021.100057

Huang, L., Zhe, T., Wu, J., Wu, Q., Pei, C., Chen, D., 2019. Robust Inter-Vehicle Distance Estimation Method Based on Monocular Vision. IEEE Access 7, 46059–46070. https://doi.org/10.1109/ACCESS.2019.2907984

Indrabayu, I., Areni, I.S., Bustamin, A., Warni, E., Tandungan, S., Irianty, R., Afifah, N.N., 2022. A Solution for Automatic Counting and Differentiate Motorcycles and Modified Motorcycles in Remote Area. Int. J. Adv. Comput. Sci. Appl. 13. https://doi.org/10.14569/IJACSA.2022.0130217

Jia, X., Tong, Y., Qiao, H., Li, M., Tong, J., Liang, B., 2023. Fast and accurate object detector for autonomous driving based on improved YOLOv5. Sci. Rep. 13, 9711. https://doi.org/10.1038/s41598-023-36868-w

Jiafa, M., Wei, H., Weiguo, S., 2020. Target distance measurement method using monocular vision. IET Image Process. 14, 3181–3187. https://doi.org/10.1049/iet-ipr.2019.1293

Kejriwal, R., H J, R., Arora, A., Mohana, 2022. Vehicle Detection and Counting using Deep Learning basedYOLO and Deep SORT Algorithm for Urban Traffic Management System, in: 2022 First International Conference on Electrical, Electronics, Information and Communication Technologies (ICEEICT). Presented at the 2022 First International Conference on Electrical, Electronics, Information and Communication Technologies (ICEEICT), IEEE, Trichy, India, pp. 1–6. https://doi.org/10.1109/ICEEICT53079.2022.9768653

Kurniawan Amiruddin, M.R., Indrabayu, Areni, I.S., Bustamin, A., 2021. An Approach for Vehicle's Classification Using BRISK Feature Extraction, in: 2021 3rd International Conference on Electronics Representation and Algorithm (ICERA). Presented at the 2021 3rd International Conference on Electronics Representation and Algorithm (ICERA), IEEE, Yogyakarta, Indonesia, pp. 83–88. https://doi.org/10.1109/ICERA53111.2021.9538701

Liu, X., Yan, W.Q., 2023. Vehicle-Related Distance Estimation Using Customized YOLOv7, in: Yan, W.Q., Nguyen, M., Stommel, M. (Eds.), Image and Vision Computing, Lecture Notes in Computer Science. Springer Nature Switzerland, Cham, pp. 91–103. https://doi.org/10.1007/978-3-031-25825-1_7

Liu, Y., Huang, H., Guo, J., Yuan, R., Xia, M., Chen, Y., 2021. Research on Target Distance Detection Technology of Vehicle Assisted driving Based on monocular Vision, in: 2021 11th International Conference on Information Technology in Medicine and Education (ITME). Presented at the 2021 11th International Conference on Information Technology in Medicine and Education (ITME), IEEE, Wuyishan, Fujian, China, pp. 17–20. https://doi.org/10.1109/ITME53901.2021.00014

Liu, Z., Chen, T., 2009. Distance Measurement System Based on Binocular Stereo Vision, in: 2009 International Joint Conference on Artificial Intelligence. Presented at the 2009 International Joint Conference on Artificial Intelligence (JCAI), IEEE, Hainan Island, China, pp. 456–459.

https://doi.org/10.1109/JCAI.2009.77
Masoumian, A., Marei, D.G.F., Abdulwahab, S., Cristiano, J., Puig, D., Rashwan, H.A., 2021. Absolute Distance Prediction Based on Deep Learning Object Detection and Monocular Depth Estimation Models, in: Villaret, M., Alsinet, T., Fernández, C., Valls, A. (Eds.), Frontiers in Artificial Intelligence and Applications. IOS Press. https://doi.org/10.3233/FAIA210151
Natanael, G., Zet, C., Fosalau, C., 2018. Estimating the distance to an object based on image processing, in: 2018 International Conference and Exposition on Electrical And Power Engineering (EPE). Presented at the 2018 International Conference and Exposition on Electrical And Power Engineering (EPE), IEEE, Iasi, pp. 0211–0216. https://doi.org/10.1109/ICEPE.2018.8559642
Nazir, S., Kaleem, M., 2023. Federated Learning for Medical Image Analysis with Deep Neural Networks. Diagnostics 13, 1532. https://doi.org/10.3390/diagnostics13091532
Neupane, B., Horanont, T., Aryal, J., 2022. Real-Time Vehicle Classification and Tracking Using a Transfer Learning-Improved Deep Learning Network.
Nguyen, H., 2019. Improving Faster R-CNN Framework for Fast Vehicle Detection. Math. Probl. Eng. 2019, 1–11. https://doi.org/10.1155/2019/3808064
Niu, C., Song, Y., Zhao, X., 2023. SE-Lightweight YOLO: Higher Accuracy in YOLO Detection for Vehicle Inspection. Appl. Sci. 13, 13052. https://doi.org/10.3390/app132413052
Oksuz, K., Cam, B.C., Kalkan, S., Akbas, E., 2021. One Metric to Measure them All: Localisation Recall Precision (LRP) for Evaluating Visual Detection Tasks.
Peixoto, B., Pinto, R., Melo, M., Cabral, L., Bessa, M., 2021. Immersive Virtual Reality for Foreign Language Education: A PRISMA Systematic Review. IEEE Access 9, 48952–48962. https://doi.org/10.1109/ACCESS.2021.3068858
Pranathi, A.S., Yadav, C.S., Rakshitha, C., Kumar, M.M.S., 2023. Detecting Objects and Estimating Distance using YOLO. Ind. Eng. J.
Qiao, D., Zulkernine, F., 2020. Vision-based Vehicle Detection and Distance Estimation, in: 2020 IEEE Symposium Series on Computational Intelligence (SSCI). Presented at the 2020 IEEE Symposium Series on Computational Intelligence (SSCI), IEEE, Canberra, ACT, Australia, pp. 2836–2842. https://doi.org/10.1109/SSCI47803.2020.9308364
Quan Meng, M., Guang Jiang, DianZhi Lai, Hua cui, Huansheng Song, 2020. Fast, Accurate Vehicle Detection and Distance Estimation. KSII Trans. Internet Inf. Syst. 14. https://doi.org/10.3837/tiis.2020.02.008
Quang, P.H., Thanh, P.P., Anh, T.N.V., Phi, S.V., Nhat, B.L., Trong, H.N., 2021. Vietnamese vehicles speed detection with video-based and deep learning for real-time traffic flow analysis system, in: 2021 15th International Conference on Advanced Computing and Applications (ACOMP). Presented at the 2021 15th International Conference on Advanced

Computing and Applications (ACOMP), IEEE, Ho Chi Minh City, Vietnam, pp. 62–69. https://doi.org/10.1109/ACOMP53746.2021.00015

Raajan, N.R., Ramkumar, M., Monisha, B., Jaiseeli, C., Venkatesan, S.P., 2012. Disparity Estimation from Stereo Images. Procedia Eng. 38, 462–472. https://doi.org/10.1016/j.proeng.2012.06.057

Rahmat, M.A., Indrabayu, Achmad, A., Salam, E.U., Bahrunnida, M.F.B., 2022. Stereo Camera Calibration For Autonomous Car Applications, in: 2022 2nd International Seminar on Machine Learning, Optimization, and Data Science (ISMODE). Presented at the 2022 2nd International Seminar on Machine Learning, Optimization, and Data Science (ISMODE), IEEE, Jakarta, Indonesia, pp. 567–572. https://doi.org/10.1109/ISMODE56940.2022.10180933

Rahul, Nair, B.B., 2018. Camera-Based Object Detection, Identification and Distance Estimation, in: 2018 2nd International Conference on Micro-Electronics and Telecommunication Engineering (ICMETE). Presented at the 2018 2nd International Conference on Micro-Electronics and Telecommunication Engineering (ICMETE), IEEE, Ghaziabad, India, pp. 203–205. https://doi.org/10.1109/ICMETE.2018.00052

Rajanna, Agarwal, R., Simha, J.B., 2022. A Monocular Camera Depth Estimate Approximation using Deep learning, in: 2022 International Conference on Futuristic Technologies (INCOFT). Presented at the 2022 International Conference on Futuristic Technologies (INCOFT), IEEE, Belgaum, India, pp. 1–4. https://doi.org/10.1109/INCOFT55651.2022.10094483

Redmon, J., Farhadi, A., 2018. YOLOv3: An Incremental Improvement. https://doi.org/10.48550/ARXIV.1804.02767

Rodríguez-Quiñonez, J.C., Sergiyenko, O., Flores-Fuentes, W., Rivas-lopez, M., Hernandez-Balbuena, D., Rascón, R., Mercorelli, P., 2017. Improve a 3D distance measurement accuracy in stereo vision systems using optimization methods' approach. Opto-Electron. Rev. 25, 24–32. https://doi.org/10.1016/j.opelre.2017.03.001

Rodríguez-Rangel, H., Morales-Rosales, L.A., Imperial-Rojo, R., Roman-Garay, M.A., Peralta-Peñuñuri, G.E., Lobato-Báez, M., 2022. Analysis of Statistical and Artificial Intelligence Algorithms for Real-Time Speed Estimation Based on Vehicle Detection with YOLO. Appl. Sci. 12, 2907. https://doi.org/10.3390/app12062907

Sadaf, M., Iqbal, Z., Javed, A.R., Saba, I., Krichen, M., Majeed, S., Raza, A., 2023. Connected and Automated Vehicles: Infrastructure, Applications, Security, Critical Challenges, and Future Aspects. Technologies 11, 117. https://doi.org/10.3390/technologies11050117

Salman, Y.D., Ku-Mahamud, K.R., Kamioka, E., 2017. DISTANCE MEASUREMENT FOR SELF-DRIVING CARS USING STEREO CAMERA.

Shen, C., Zhao, X., Liu, Z., Gao, T., Xu, J., 2020. Joint vehicle detection and distance prediction via monocular depth estimation. IET Intell. Transp. Syst. 14, 753–763. https://doi.org/10.1049/iet-its.2019.0376

Song, H., Liang, H., Li, H., Dai, Z., Yun, X., 2019. Vision-based vehicle detection

and counting system using deep learning in highway scenes. Eur. Transp. Res. Rev. 11, 51. https://doi.org/10.1186/s12544-019-0390-4

Strbac, B., Gostovic, M., Lukac, Z., Samardzija, D., 2020. YOLO Multi-Camera Object Detection and Distance Estimation, in: 2020 Zooming Innovation in Consumer Technologies Conference (ZINC). Presented at the 2020 Zooming Innovation in Consumer Technologies Conference (ZINC), IEEE, Novi Sad, Serbia, pp. 26–30. https://doi.org/10.1109/ZINC50678.2020.9161805

Tao, N., Xiangkun, J., Xiaodong, D., Jinmiao, S., Ranran, L., 2022. Vehicle detection method with low-carbon technology in haze weather based on deep neural network. Int. J. Low-Carbon Technol. 17, 1151–1157. https://doi.org/10.1093/ijlct/ctac084

Thu Hien, N.T., Hien, T.T., Le Dinh Chung, Tien Dzung Nguyen, 2023. Deep Learning Based Vehicle Speed Estimation on Highways. JST Smart Syst. Devices 33, 43–53. https://doi.org/10.51316/jst.163.ssad.2023.33.1.6

Tian, J., Jin, Q., Wang, Y., Yang, J., Zhang, S., Sun, D., 2024. Performance analysis of deep learning-based object detection algorithms on COCO benchmark: a comparative study. J. Eng. Appl. Sci. 71, 76. https://doi.org/10.1186/s44147-024-00411-z

Ting-Na Liu, T.-N.L., Ting-Na Liu, Z.-J.Z., Zhong-Jie Zhu, Y.-Q.B., Yong-Qiang Bai, G.-L.L., Guang-Long Liao, Y.-X.C., 2022. YOLO-Based Efficient Vehicle Object Detection. 電腦學刊 33, 069–079. https://doi.org/10.53106/199115992022083304006

Trifu, A., Smîdu, E., Badea, D.O., Bulboacă, E., Haralambie, V., 2022. Applying the PRISMA method for obtaining systematic reviews of occupational safety issues in literature search. MATEC Web Conf. 354, 00052. https://doi.org/10.1051/matecconf/202235400052

Vajgl, M., Hurtik, P., Nejezchleba, T., 2022. Dist-YOLO: Fast Object Detection with Distance Estimation. Appl. Sci. 12, 1354. https://doi.org/10.3390/app12031354

Wang, B.-L., King, C.-T., Chu, H.-K., 2018. A Semi-Automatic Video Labeling Tool for Autonomous Driving Based on Multi-Object Detector and Tracker, in: 2018 Sixth International Symposium on Computing and Networking (CANDAR). Presented at the 2018 Sixth International Symposium on Computing and Networking (CANDAR), IEEE, Takayama, pp. 201–206. https://doi.org/10.1109/CANDAR.2018.00035

Wang, H., Yu, Yi., Cai, Y., Chen, X., Chen, L., Liu, Q., 2019. A Comparative Study of State-of-the-Art Deep Learning Algorithms for Vehicle Detection. IEEE Intell. Transp. Syst. Mag. 11, 82–95. https://doi.org/10.1109/MITS.2019.2903518

Wang, J., Dong, Y., Zhao, S., Zhang, Z., 2023. A High-Precision Vehicle Detection and Tracking Method Based on the Attention Mechanism. Sensors 23, 724. https://doi.org/10.3390/s23020724

Wang, Y., Guan, Y., Liu, H., Jin, L., Li, X., Guo, B., Zhang, Z., 2023. VV-YOLO: A Vehicle View Object Detection Model Based on Improved YOLOv4. Sensors 23, 3385. https://doi.org/10.3390/s23073385

Wang, Y., Xu, S., Wang, P., Li, K., Song, Z., Zheng, Q., Li, Y., He, Q., 2024. Lightweight Vehicle Detection Based on Improved YOLOv5s. Sensors 24, 1182. https://doi.org/10.3390/s24041182

Xie, F., Yin, P., Ke, Z., Sun, R., Ding, K., Chen, J., Wu, Q., 2021. Vehicle Segmentation with Coarse Distance Estimation Based on Monocular Vision, in: 2021 IEEE International Conference on E-Business Engineering (ICEBE). Presented at the 2021 IEEE International Conference on e-Business Engineering (ICEBE), IEEE, Guangzhou, China, pp. 16–20. https://doi.org/10.1109/ICEBE52470.2021.00019

Xú, G., Zhang, Z., 1996. Epipolar geometry in stereo, motion, and object recognition: a unified approach, Computational imaging and vision. Kluwer Academic Publishers, Dordrecht ; Boston.

Yang, R., Yu, S., Yao, Q., Huang, J., Ya, F., 2023. Vehicle Distance Measurement Method of Two-Way Two-Lane Roads Based on Monocular Vision. Appl. Sci. 13, 3468. https://doi.org/10.3390/app13063468

Zaarane, A., Slimani, I., Al Okaishi, W., Atouf, I., Hamdoun, A., 2020. Distance measurement system for autonomous vehicles using stereo camera. Array 5, 100016. https://doi.org/10.1016/j.array.2020.100016

Zelinsky, A., 2009. Learning OpenCV---Computer Vision with the OpenCV Library (Bradski, G.R. et al.; 2008)[On the Shelf]. IEEE Robot. Autom. Mag. 16, 100–100. https://doi.org/10.1109/MRA.2009.933612

Zhang, F., Jin, Y., Kan, S., Zhang, L., Cen, Y., Jin, W., 2021. Distorted Vehicle Detection and Distance Estimation by Metric Learning-Based SSD: Int. J. Comput. Intell. Syst. 14, 1426. https://doi.org/10.2991/ijcis.d.210419.001

Zhang, Y., Song, X., Bai, B., Xing, T., Liu, C., Gao, X., Wang, Z., Wen, Y., Liao, H., Zhang, G., Xu, P., 2021. 2nd Place Solution for Waymo Open Dataset Challenge -- Real-time 2D Object Detection. https://doi.org/10.48550/ARXIV.2106.08713

Zhang, Z., 2000. A flexible new technique for camera calibration. IEEE Trans. Pattern Anal. Mach. Intell. 22, 1330–1334. https://doi.org/10.1109/34.888718

Zhe, T., Huang, L., Wu, Q., Zhang, J., Pei, C., Li, L., 2020. Inter-Vehicle Distance Estimation Method Based on Monocular Vision Using 3D Detection. IEEE Trans. Veh. Technol. 69, 4907–4919. https://doi.org/10.1109/TVT.2020.2977623

Zou, Z., Chen, K., Shi, Z., Guo, Y., Ye, J., 2023. Object Detection in 20 Years: A Survey.

Lampiran

Source code general.py

```
import cv2
import os
import yaml

f = open('./config.yaml')
config = yaml.safe_load(f)
f.close()

# Header template
def unfoldHeader(cls):
    if cls:
        os.system('cls')
    else:
        print('\n')
    print('\033[92m``Program Disertasi``\033[0m v.1.3.0')
    print('\033[1mDeteksi Jarak Relatif Kendaraan.\033[0m')
    print('\n\033[1mMuhammad Abdillah Rahmat - D053202001\033[0m')
    print('\033[1mGowa - Makassar, 2024\033[0m')
    print('\n----------------------------------------------------------------
-----------------\n')

    # Instruction for brightness & contrast
    print('=== CAMERA CONTROLS FOR BRIGHTNESS & CONTRAST ===')
    # print('\nPress "]" to turn up the brightness.')
    # print('Press "[" to turn down the brightness.\n')
    # print('Press "." to turn up the contrast.')
    # print('Press "," to turn down the contrast.\n')
    #
    # if(config['capture']['mode'] != 'video'):
    #     print('Press', "\"'\"", "to turn up the exposure.")
    #     print('Press ";" to turn down the exposure.\n')

    input('Press "ENTER" key to continue.\n\n')

# Template for error message
def errorMessage(msg):
    print("\n\033[91mERRRRRR!!\033[0m")
    print("Message: " + msg)
    quit()

# Check original dimension
def originalDimCheck(cam):
    if config['capture']['mode'] == 'video':
        return cam.get(cv2.CAP_PROP_FRAME_WIDTH),
cam.get(cv2.CAP_PROP_FRAME_HEIGHT), cam.get(cv2.CAP_PROP_FRAME_WIDTH),
cam.get(cv2.CAP_PROP_FRAME_HEIGHT)

    return round(cam.get_camera_information().camera_resolution.width,
2), round(cam.get_camera_information().camera_resolution.height, 2),
round(cam.get_camera_information().camera_resolution.width, 2),
round(cam.get_camera_information().camera_resolution.height, 2)

# Template error related from detection
def errorDetection(msg, frameL, frameR):
    print("\nERRRR: " + msg)
    return frameL, frameR
```

Source code main.py

```
# Import libraries and functions
import pandas as pd
import cv2
import torch
import yaml
import os
import pyzed.sl as sl
import time
import argparse
import numpy as np

from helper.general import unfoldHeader, errorMessage, errorDetection
from helper.load import zedStereo, stereoCamera
from helper.distance import convertBbox, bboxLabelDistance
from helper.rmse import saveData

# Initialize argument parser
parser = argparse.ArgumentParser(description='Stereo Camera Object
Detection and Speed Estimation')
parser.add_argument('--video', type=str, help='Path to the video file')
args = parser.parse_args()

# Determine the source of input
use_video = args.video is not None
video_path = args.video if use_video else ""

###### LOAD YAML ######

f = open('config.yaml')
dataConfig = yaml.safe_load(f)
f.close()

# ``unfold`` Header
unfoldHeader(dataConfig['header']['cls'])

# Load data from yaml
if dataConfig['cameraConfig']['model']:
    model_custom = './models/' + dataConfig['cameraConfig']['model']
else:
    model_custom = './models/' + 'best.pt'

conf_custom = dataConfig['cameraConfig']['conf'] if
dataConfig['cameraConfig']['conf'] else 0
iou_custom = dataConfig['cameraConfig']['iou'] if
dataConfig['cameraConfig']['iou'] else 0

mode_rmse = dataConfig['rmse']['mode'] if dataConfig['rmse']['mode'] else
False
frame_rmse = dataConfig['rmse']['maxFramesPerDist'] if 'maxFramesPerDist'
in dataConfig['rmse'] else 0
dist_rmse = dataConfig['rmse']['setDistance'] if 'setDistance' in
dataConfig['rmse'] else 0
result_rmse = {}

mode_capture = dataConfig['capture']['mode']

if mode_capture == 'video' and dataConfig['capture']['cam1'] and
dataConfig['capture']['cam2']:
    cam1_capture = dataConfig['capture']['cam1']
    cam2_capture = dataConfig['capture']['cam2']
```

```
elif mode_capture == 'video':
    _, _ = errorMessage("Cam 1/Cam 2 source video contains false!")
    quit()

###### END OF LOAD YAML ######

###### LOAD STEREO CAMERA OR VIDEO FILE ######

if use_video:
    print(f"=== LOAD VIDEO FILE: {video_path} ===")
    cap = cv2.VideoCapture(video_path)
    if not cap.isOpened():
        print(f"Error: Cannot open video file {video_path}")
        quit()
    widthL = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
    heightL = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
    widthR = widthL  # Assuming both left and right videos have same
dimensions
    heightR = heightL
    alpha = dataConfig['cameraConfig']['contrast'] if
dataConfig['cameraConfig']['contrast'] else 1.0
    beta = dataConfig['cameraConfig']['brightness'] if
dataConfig['cameraConfig']['brightness'] else 0
else:
    if mode_capture == 'video':
        print("=== LOAD VIDEO ===")
        inputL = os.getcwd() + '\\video\\' + cam1_capture
        inputR = os.getcwd() + '\\video\\' + cam2_capture
        camL, camR, widthL, heightL, widthR, heightR =
stereoCamera(inputL, inputR, False)
        alpha = dataConfig['cameraConfig']['contrast'] if
dataConfig['cameraConfig']['contrast'] else 1.0
        beta = dataConfig['cameraConfig']['brightness'] if
dataConfig['cameraConfig']['brightness'] else 0
    else:
        print("=== LOAD STEREO CAMERA ===")
        cam, runtime, widthL, heightL, widthR, heightR = zedStereo()
        brightness = -1
        contrast = -1
        exposure = -1

dim = (widthL, heightL)

###### END OF LOAD STEREO CAMERA OR VIDEO FILE ######

###### LOAD YOLO ######

print("\n\n=== RUNNING YOLO ===\n")
try:
    model = torch.hub.load('yolov5-detect', 'custom', path=model_custom,
source='local')
except Exception as e:
    print(e)
    errorMessage("Cannot load model, please check 'torch.hub.load'
function!")

###### END OF LOAD YOLO ######

# Function to calculate speed
def calculate_speed(previous_positions, previous_times,
current_positions, current_time):
    speeds = {}
    for vehicle_id, current_position in current_positions.items():
```

```
        if vehicle_id in previous_positions:
            delta_distance = current_position -
previous_positions[vehicle_id]
            delta_time = current_time - previous_times[vehicle_id]
            if delta_time > 0:
                speed = delta_distance / delta_time
            else:
                speed = 0
            speeds[vehicle_id] = speed
        else:
            speeds[vehicle_id] = 0  # First detection, speed is zero
        previous_positions[vehicle_id] = current_position
        previous_times[vehicle_id] = current_time
    return speeds

# Function to measure stereoscopic distance
def stereoscopicMeasurement(leftX, rightX, width, baseline,
focal_length):
    disparity = abs(leftX - rightX)
    if disparity == 0:
        return np.inf
    distance = (focal_length * baseline) / disparity
    return distance, disparity

###### RUN YOLO TO OpenCV ######

print("\n\n=== PUT YOLO INTO STEREO CAMERA ===")
print("=== APPLY DISTANCE MEASUREMENT ===")
initial_frame = 0

# Initialize speed tracking data
previous_positions = {}
previous_times = {}

while True:
    # Start timing the computation
    start_time = time.time()

    if mode_rmse:
        print("\nFrame: " + str(initial_frame))
        if initial_frame == frame_rmse:
            saveData(dist_rmse, result_rmse)
            break

    classes = list()
    distances = list()
    speeds = list()  # To store speeds of detected vehicles

    try:
        ###### STEREO CAMERA & SETTINGS ######

        model.conf = conf_custom
        model.iou = iou_custom
        if dataConfig['cameraConfig']['customModel'] != False:
            model.classes = dataConfig['cameraConfig']['customModel']

        if use_video:
            ret, frame = cap.read()
            if not ret:
                break
            result_left = frame
            result_right = frame  # For simplicity, using the same frame
as left and right input
```

```
            result_left = cv2.convertScaleAbs(result_left, alpha=alpha,
beta=beta)
            result_right = cv2.convertScaleAbs(result_right, alpha=alpha,
beta=beta)
        elif mode_capture == 'video':
            retL, result_left = camL.read()
            retR, result_right = camR.read()
            if retL == False or retR == False:
                break
            result_left = cv2.convertScaleAbs(result_left, alpha=alpha,
beta=beta)
            result_right = cv2.convertScaleAbs(result_right, alpha=alpha,
beta=beta)
        else:
            left_image = sl.Mat()
            right_image = sl.Mat()
            err = cam.grab(runtime)
            if err == sl.ERROR_CODE.SUCCESS:
                cam.set_camera_settings(sl.VIDEO_SETTINGS.BRIGHTNESS,
brightness)
                cam.set_camera_settings(sl.VIDEO_SETTINGS.CONTRAST,
contrast)
                cam.set_camera_settings(sl.VIDEO_SETTINGS.EXPOSURE,
exposure)
                cam.retrieve_image(left_image, sl.VIEW.LEFT)
                result_left = left_image.get_data()
                cam.retrieve_image(right_image, sl.VIEW.RIGHT)
                result_right = right_image.get_data()
            else:
                errorMessage("Something wrong with ZED Stereo Camera")

        frameGrayL = cv2.cvtColor(result_left, cv2.COLOR_BGR2GRAY)
        frameGrayR = cv2.cvtColor(result_right, cv2.COLOR_BGR2GRAY)
        frameGrayL = cv2.equalizeHist(frameGrayL)
        frameGrayR = cv2.equalizeHist(frameGrayR)

        key = cv2.waitKey(10)
        resultLR = model([frameGrayL], augment=True)

        if mode_capture == 'live':
            if key == ord(']'):
                if brightness == 8:
                    print("=== Brightness mencapai maks! ===")
                else:
                    brightness += 1
            elif key == ord('['):
                if brightness == -1:
                    print("=== Brightness dalam mode auto ===")
                else:
                    brightness -= 1
            if key == ord('.'):
                if contrast == 8:
                    print("=== Contrast mencapai maks! ===")
                else:
                    contrast += 1
            elif key == ord(','):
                if contrast == -1:
                    print("=== Contrast dalam mode auto ===")
                else:
                    contrast -= 1
            if key == ord("'"):
                if exposure == 100:
                    print("=== Exposure mencapai maks!")
```

```
                elif exposure == -1:
                    exposure = 0
                else:
                    exposure += 10
            elif key == ord(';'):
                if exposure == -1:
                    print("=== Exposure dalam mode auto!")
                elif exposure == 0:
                    exposure = -1
                else:
                    exposure -= 10
        else:
            if key == ord(']'):
                if alpha == 3.0:
                    print("=== Brightness mencapai maks! ===")
                else:
                    alpha += 0.5
                    print(f"Brightness: {alpha}")
            elif key == ord('['):
                if alpha == 1.0:
                    print("=== Brightness dalam mode auto ===")
                else:
                    alpha -= 0.5
                    print(f"Brightness: {alpha}")
            if key == ord('.'):
                if beta == 100:
                    print("=== Contrast mencapai maks! ===")
                else:
                    beta += 10
                    print(f"Contrast: {beta}")
            elif key == ord(','):
                if beta == 0:
                    print("=== Contrast dalam mode auto ===")
                else:
                    beta -= 10
                    print(f"Contrast: {beta}")

        ###### MATCH TEMPLATE ######

        labelL = resultLR.pandas().xyxy[0]
        labelR = pd.DataFrame({})

        for i in range(len(labelL)):
            image =
frameGrayL[int(labelL.iloc[i]['ymin']):int(labelL.iloc[i]['ymax']),
int(labelL.iloc[i]['xmin']):int(labelL.iloc[i]['xmax'])]
            height, width = image.shape[::]
            match = cv2.matchTemplate(frameGrayR, image, cv2.TM_SQDIFF)
            _, _, minloc, maxloc = cv2.minMaxLoc(match)
            data = {
                "xmin": float(minloc[0]),
                "ymin": float(minloc[1]),
                "xmax": float(minloc[0] + width),
                "ymax": float(minloc[1] + height),
                "confidence": labelL.iloc[i]['confidence'],
                "class": labelL.iloc[i]['class'],
                "name": labelL.iloc[i]['name']
            }
            labelR = pd.concat([labelR, pd.DataFrame(data, index=[i])])

        ###### END OF MATCH TEMPLATE ######

        ###### END OF STEREO CAMERA & SETTINGS ######
```

```
        ###### PRINT INTO COMMAND PROMPT ######

        print("\n-----------------------------------------")

        if len(labelL) and len(labelR):
            labelR = labelR.sort_values(by=['confidence'],
ascending=False)
            if len(labelL) == len(labelR):
                current_positions = {}
                for i in range(len(labelL)):
                    labelL.at[i, 'name'] = labelL.iloc[i]['name'] +
str(i)
                    labelR.at[i, 'name'] = labelR.iloc[i]['name'] +
str(i)

                print("\nDetection on Left Camera: ")
                print(labelL)
                print("\nDetection on Right Camera (from template
matching): ")
                print(labelR)

                id = 0
                while id < len(labelL):
                    xl, yl, wl, hl =
convertBbox(round(labelL.iloc[id]['xmin'],
dataConfig['cameraConfig']['detectRound']),
round(labelL.iloc[id]['ymin'],
dataConfig['cameraConfig']['detectRound']),
round(labelL.iloc[id]['xmax'],
dataConfig['cameraConfig']['detectRound']),
round(labelL.iloc[id]['ymax'],
dataConfig['cameraConfig']['detectRound']))
                    xr, yr, wr, hr = convertBbox(labelR.iloc[id]['xmin'],
labelR.iloc[id]['ymin'], labelR.iloc[id]['xmax'],
labelR.iloc[id]['ymax'])

                    if dataConfig['cameraConfig']['blockDiffClass']:
                        if labelL.iloc[id]['name'] ==
labelR.iloc[id]['name']:
                            print("\n\nx1 for left camera = " + str(xl))
                            print("x2 for right camera = " + str(xr))

                            distance, disparity =
stereoscopicMeasurement(xl, xr, dim[0],
dataConfig['cameraConfig']['baseline'],
dataConfig['cameraConfig']['focal_length'])

                            classes.append(labelL.iloc[id]['name'])
                            distances.append(distance)

                            current_positions[labelL.iloc[id]['name']] =
distance

                            # Append distance into RMSE
                            if mode_rmse:
                                if not labelL.iloc[id]['name'] in
result_rmse:
                                    result_rmse[labelL.iloc[id]['name']]
= list()
                                    result_rmse["disp_" +
labelL.iloc[id]['name']] = list()
```

```
result_rmse[labelL.iloc[id]['name']].append(round(distance,
dataConfig['rmse']['distRound']))
                                result_rmse["disp_" +
labelL.iloc[id]['name']].append(round(disparity,
dataConfig['rmse']['distRound']))
                        else:
                            resultImgL, resultImgR =
errorDetection("Class Left & Right is not same!", result_left,
result_right)
                            break
                    else:
                        print("\n\nx1 for left camera = " + str(xl))
                        print("x2 for right camera = " + str(xr))

                        distance, disparity = stereoscopicMeasurement(xl,
xr, dim[0], dataConfig['cameraConfig']['baseline'],
dataConfig['cameraConfig']['focal_length'])
                        classes.append(labelL.iloc[id]['name'])
                        distances.append(distance)

                        current_positions[labelL.iloc[id]['name']] =
distance

                        # Append distance into RMSE
                        if mode_rmse:
                            if not labelL.iloc[id]['name'] in
result_rmse:
                                result_rmse[labelL.iloc[id]['name']] =
list()
                                result_rmse["disp_" +
labelL.iloc[id]['name']] = list()

result_rmse[labelL.iloc[id]['name']].append(distance)
                            result_rmse["disp_" +
labelL.iloc[id]['name']].append(round(disparity,
dataConfig['rmse']['distRound']))

                    id += 1

                current_time = time.time()
                speeds = calculate_speed(previous_positions,
previous_times, current_positions, current_time)

                initial_frame += 1

                if len(classes):
                    data = {
                        'class': classes,
                        'distance': distances,
                        'speed': list(speeds.values())
                    }

                    data = pd.DataFrame(data)

                    print("\nDistance and Speed Measurement:")
                    print(data)

                    # Put manual bbox and distance in frame
                    resultImgL = bboxLabelDistance(labelL, data,
result_left)
                    resultImgR = bboxLabelDistance(labelR, data,
result_right)
            else:
```

```
                resultImgL, resultImgR = errorDetection("Total label in L
doesn't same as total label in R", result_left, result_right)
        else:
            resultImgL, resultImgR = errorDetection("No detection on
left/right camera!", result_left, result_right)

        ###### END OF PRINT TO COMMAND PROMPT ######

        ###### SHOW CAMERAS IN REALTIME ######

        if dataConfig['cameraConfig']['combinedCamera']:
            alphaCombined = 0.5
            betaCombined = 1.0
            combineImg = cv2.addWeighted(resultImgR, alphaCombined,
resultImgL, betaCombined, 0.0)

            if dataConfig['cameraConfig']['resolution'] == 'HD720' or
dataConfig['cameraConfig']['resolution'] == 'HD1080':
                combineImg = cv2.resize(combineImg, (672, 376))
            cv2.imshow("Combined Cameras", combineImg)
        else:
            if dataConfig['cameraConfig']['resolution'] == 'HD720' or
dataConfig['cameraConfig']['resolution'] == 'HD1080':
                resultImgL = cv2.resize(resultImgL, (672, 376))
                resultImgR = cv2.resize(resultImgR, (672, 376))
            cv2.imshow("Left Camera", resultImgL)
            cv2.imshow("Right Camera", resultImgR)

        ###### END OF SHOW CAMERAS IN REALTIME ######

        # End timing the computation
        end_time = time.time()
        computation_time = end_time - start_time
        print(f"Computation Time for this frame: {computation_time:.4f}
seconds")

        # Key to exit
        if key == ord('q') or key == ord('Q'):
            if mode_rmse:
                saveData(dist_rmse, result_rmse)
            print("\n\nExited!")
            break

    except KeyboardInterrupt:
        if mode_rmse:
            saveData(dist_rmse, result_rmse)
        print("\n\nExited!")
        break

###### END OF RUN YOLOv5 TO OpenCV ######

cv2.destroyAllWindows()
print("\nThank you!\n:)")
```

Source code distance.py

```
import math
import cv2
import yaml
from helper.general import errorMessage

# Import config
config = yaml.safe_load(open('./config.yaml'))

# Create manual bbox label and distance
def bboxLabelDistance(dataBbox, data, frame):
    i = 0
    while i < len(data):
        label = data.iloc[i]['class']
        distance = round(data.iloc[i]['distance'], 2)
        speed = round(data.iloc[i]['speed'], 2)

        if distance < config['distanceConfig']['min']:
            distance_text = 'Too close!'
        elif distance > config['distanceConfig']['max']:
            distance_text = 'Too far!'
        else:
            distance_text = f"{distance}m"

        speed_text = f"{speed}m/s"

        xmin = int(dataBbox.iloc[i]['xmin'])
        ymin = int(dataBbox.iloc[i]['ymin'])
        xmax = int(dataBbox.iloc[i]['xmax'])
        ymax = int(dataBbox.iloc[i]['ymax'])

        if config['cameraConfig']['resolution'] == "HD1080":
            text_size, _ = cv2.getTextSize(label + ' ' + distance_text +
' ' + speed_text, cv2.FONT_HERSHEY_SIMPLEX, 3, 4)
            text_w, text_h = text_size

            resultImg = cv2.rectangle(frame, (xmin, ymin), (xmax, ymax),
(0, 255, 0), 2)
            resultImg = cv2.rectangle(frame, (xmin, ymin), (xmin +
text_w, ymin - text_h), (0, 0, 0), -1)
            resultImg = cv2.putText(frame, label + ': ' + distance_text +
' ' + speed_text, (xmin, ymin-5), cv2.FONT_HERSHEY_SIMPLEX, 2, (255, 255,
255), 2, cv2.LINE_AA)
        elif config['cameraConfig']['resolution'] == "HD720":
            text_size, _ = cv2.getTextSize(label + ' ' + distance_text +
' ' + speed_text, cv2.FONT_HERSHEY_SIMPLEX, 2, 3)
            text_w, text_h = text_size

            resultImg = cv2.rectangle(frame, (xmin, ymin), (xmax, ymax),
(0, 255, 0), 2)
            resultImg = cv2.rectangle(frame, (xmin, ymin), (xmin +
text_w, ymin - text_h), (0, 0, 0), -1)
            resultImg = cv2.putText(frame, label + ': ' + distance_text +
' ' + speed_text, (xmin, ymin-5), cv2.FONT_HERSHEY_SIMPLEX, 1.5, (255,
255, 255), 2, cv2.LINE_AA)
        elif config['cameraConfig']['resolution'] == "VGA":
            text_size, _ = cv2.getTextSize(label + ' ' + distance_text +
' ' + speed_text, cv2.FONT_HERSHEY_SIMPLEX, 1, 2)
            text_w, text_h = text_size

            resultImg = cv2.rectangle(frame, (xmin, ymin), (xmax, ymax),
```

```
(0, 255, 0), 2)
            resultImg = cv2.rectangle(frame, (xmin, ymin), (xmin +
text_w, ymin - text_h), (0, 0, 0), -1)
            resultImg = cv2.putText(frame, label + ': ' + distance_text +
' ' + speed_text, (xmin, ymin-5), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (255,
255, 255), 1, cv2.LINE_AA)
        i += 1

    return resultImg

# Converting the results from PyTorch hub
def convertBbox(x1, y1, x2, y2):
    xc = (x1 + x2) / 2
    yc = (y1 + y2) / 2
    width = (x2 - x1)
    height = (y2 - y1)

    return xc, yc, width, height

# Stereoscopic Measurement
"""
PARAMETERS:

leftX = target coordinates in the x-axis for left camera (px)
rightX = target coordinates in the x-axis for right camera (px)
width = width taken from image dimension (px)
b = baseline (actual distance between two cameras) (m)
fov = field of view/lens view angle (two cameras must be of the same
model)
"""

# kamera zed (person) - 2.0
# kamera zed (motorcycle) - 3.0
# kamera zed (car) - 1.5
def stereoscopicMeasurement(leftX, rightX, width, b, fov):
    baselineWidth = float(b) * float(width)
    disparity = round(abs(float(leftX) - float(rightX)),
config['cameraConfig']['detectRound'])
    fieldOfView = float(math.tan(math.radians(fov / 2)))

    print("\nBaseline x width: " + str(baselineWidth))
    print("Disparity: " + str(disparity))
    print("Field of View: " + str(fieldOfView))

    try:
        distance = baselineWidth / ((2 * fieldOfView) * disparity)
    except ZeroDivisionError:
        distance = 0

    return distance, disparity
```

Source code load.py

```
import cv2
import os
import yaml
import pyzed.sl as sl
from helper.general import originalDimCheck, errorMessage

f = open('./config.yaml')
config = yaml.safe_load(f)
f.close()

def zedStereo():
    init = sl.InitParameters()

    # Change resolution & FPS
(https://www.stereolabs.com/docs/video/camera-controls/#using-the-api)

    if(config['cameraConfig']['resolution'] == "VGA"):
        init.camera_resolution = sl.RESOLUTION.VGA
    elif(config['cameraConfig']['resolution'] == "HD720"):
        init.camera_resolution = sl.RESOLUTION.HD720
    elif(config['cameraConfig']['resolution'] == "HD1080"):
        init.camera_resolution = sl.RESOLUTION.HD1080
    else:
        errorMessage("Cannot open webcam!")

    init.camera_fps = 60

    # End of change resolution & FPS



    runtime = sl.RuntimeParameters()

    cam = sl.Camera()

    if not cam.is_opened:
        errorMessage("Cannot open webcam!")

    status = cam.open(init)

    if status != sl.ERROR_CODE.SUCCESS:
        errorMessage("Cannot open webcam!")

    # Show original dimension
    widthL, heightL, widthR, heightR = originalDimCheck(cam)

    print("\nVideo 1 original dimension: " + str(widthL) + ' ' +
str(heightL))
    print("Video 2 original dimension: " + str(widthR) + ' ' +
str(heightR))

    print("\nSuccess: Stereo Camera successfully loaded!")

    return cam, runtime, widthL, heightL, widthR, heightR

def stereoCamera(L, R, dshow):
    if dshow:
        camL = cv2.VideoCapture(L, cv2.CAP_DSHOW)
        camR = cv2.VideoCapture(R, cv2.CAP_DSHOW)
    else:
```

```
            camL = cv2.VideoCapture(L)
            camR = cv2.VideoCapture(R)

        if not camL.isOpened() & camR.isOpened():
            errorMessage("Cannot open webcam!")

        # Show original dimension
        widthL, heightL, widthR, heightR = originalDimCheck(camL)

        print("\nVideo 1 original dimension: " + str(widthL) + ' ' +
str(heightL))
        print("Video 2 original dimension: " + str(widthR) + ' ' +
str(heightR))

        print("\nSuccess: Stereo Camera successfully loaded!")

        return camL, camR, widthL, heightL, widthR, heightR
```

Tabel 7. Skenario 1 Deteksi Jarak lengkap (Stereo dan Mono Camera)

Vehicle	***Actual Distance (m)***	***Prediction Stereo (m)***	***Absolute Percent Error Stereo (%)***	***Prediction Mono (m)***	***Absolute Percent Error Mono (%)***
Motorcycle	5	4.68	6.40	4.82	3.60
Motorcycle	10	9.53	4.70	9.73	2.70
Motorcycle	15	17.47	16.47	17.12	14.13
Motorcycle	20	22.58	12.90	23.12	15.60
Motorcycle	25	28.12	12.48	27.44	9.76
Motorcycle	30	34.33	14.43	32.20	7.33
Car	5	4.74	5.20	4.90	2.00
Car	10	10.23	2.30	9.82	1.80
Car	15	17.22	14.80	16.11	7.27
Car	20	22.77	13.85	21.21	5.95
Car	25	27.87	11.48	27.20	8.80
Car	30	33.26	10.87	32.12	6.27
MAPE			**10.24**		**7.09**

Tabel 8. Skenario 2 Deteksi Jarak lengkap (Stereo dan Mono Camera)

Vehicle	***Actual Distance (m)***	***Prediction Stereo (m)***	***Absolute Percent Error Stereo (%)***	***Prediction Mono (m)***	***Absolute Percent Error Mono (%)***
Motorcycle	5	4.86	2.80	4.91	1.79
Motorcycle	10	9.89	1.10	9.91	0.89
Motorcycle	15	15.23	1.53	15.45	2.99
Motorcycle	20	22.25	11.25	22.11	10.55
Motorcycle	25	27.31	9.24	27.14	8.56
Motorcycle	30	32.23	7.43	31.60	5.33
Car	5	5.19	3.80	4.90	1.99
Car	10	9.94	0.60	9.82	1.79
Car	15	15.42	2.80	16.11	7.40
Car	20	22.12	10.60	21.21	6.05
Car	25	26.92	7.68	27.20	8.80
Car	30	32.11	7.03	32.12	7.06
MAPE			**5.48**		**5.27**