(12) United States Patent
Farabet et al.

(10) **Patent No.:** **US 12,182,694 B2**

(45) **Date of Patent:** **Dec. 31, 2024**

(54) **TRAINING, TESTING, AND VERIFYING AUTONOMOUS MACHINES USING SIMULATED ENVIRONMENTS**

(71) Applicant: **NVIDIA Corporation**, Santa Clara, CA (US)

(72) Inventors: **Clement Farabet**, Mill Valley, CA (US); **John Zedlewski**, San Francisco, CA (US); **Zachary Taylor**, Santa Cruz, CA (US); **Greg Heinrich**, Nice (FR); **Claire Delaunay**, Menlo Park, CA (US); **Mark Daly**, Eagle, ID (US); **Matthew Campbell**, Surf City, NC (US); **Curtis Beeson**, Irwin, PA (US); **Gary Hicok**, Mesa, CA (US); **Michael Cox**, Menlo Park, CA (US); **Rev Lebaredian**, Los Gatos, CA (US); **Tony Tamasi**, Portola Valley, CA (US); **David Auld**, Saratoga, CA (US)

(73) Assignee: **NVIDIA Corporation**, Santa Clara, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **17/898,887**

(22) Filed: **Aug. 30, 2022**

(65) **Prior Publication Data**

US 2023/0004801 A1 Jan. 5, 2023

Related U.S. Application Data

(63) Continuation of application No. 16/366,875, filed on Mar. 27, 2019, now Pat. No. 11,436,484.

(Continued)

(51) **Int. Cl.**
G06F 9/455	(2018.01)
G06F 18/2413	(2023.01)

(Continued)

(52) **U.S. Cl.**
CPC .... *G06N 3/063* (2013.01); *G06F 9/455* (2013.01); *G06F 18/24133* (2023.01); (Continued)

(58) **Field of Classification Search**
CPC G06N 3/063; G06N 20/00; G06N 3/045; G06N 3/08; G06V 10/454; G06V 10/82; G06V 20/56; G06F 18/24133; G06F 9/455
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

7,409,295	B2	8/2008	Paradie
8,204,642	B2	6/2012	Tanaka et al.

(Continued)

FOREIGN PATENT DOCUMENTS

CN	103677838 A	3/2014
DE	102015221920 A1	5/2017

(Continued)

OTHER PUBLICATIONS

Towards End-to-End Lane Detection: an Instance Segmentation Approach. Neven et al. (Year: 2018).
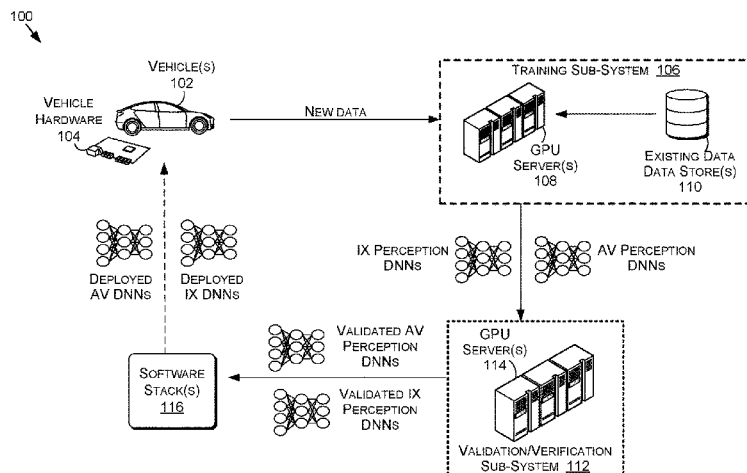
(Continued)

Primary Examiner — Beau D Spratt
(74) *Attorney, Agent, or Firm* — Taylor English Duma L.L.P.

(57) **ABSTRACT**

In various examples, physical sensor data may be generated by a vehicle in a real-world environment. The physical sensor data may be used to train deep neural networks (DNNs). The DNNs may then be tested in a simulated environment—in some examples using hardware configured for installation in a vehicle to execute an autonomous driving software stack—to control a virtual vehicle in the simulated environment or to otherwise test, verify, or vali-

(Continued)

date the outputs of the DNNs. Prior to use by the DNNs, virtual sensor data generated by virtual sensors within the simulated environment may be encoded to a format consistent with the format of the physical sensor data generated by the vehicle.

20 Claims, 23 Drawing Sheets

Related U.S. Application Data

(60) Provisional application No. 62/648,399, filed on Mar. 27, 2018.

(51) **Int. Cl.**

G06N 3/045	(2023.01)
G06N 3/063	(2023.01)
G06N 3/08	(2023.01)
G06N 20/00	(2019.01)
G06V 10/44	(2022.01)
G06V 10/764	(2022.01)
G06V 10/82	(2022.01)
G06V 20/56	(2022.01)

(52) **U.S. Cl.**
 CPC *G06N 3/045* (2023.01); *G06N 3/08* (2013.01); *G06N 20/00* (2019.01); *G06V 10/454* (2022.01); *G06V 10/764* (2022.01); *G06V 10/82* (2022.01); *G06V 20/56* (2022.01)

(56) **References Cited**

U.S. PATENT DOCUMENTS

9,373,057	B1	6/2016	Erhan et al.
9,701,307	B1	7/2017	Newman et al.
10,007,269	B1*	6/2018	Gray G06F 18/2178
10,108,867	B1*	10/2018	Vallespi-Gonzalez G06V 20/58
10,133,274	B2	11/2018	Shashua et al.
10,134,278	B1	11/2018	Konrardy et al.
10,157,331	B1	12/2018	Tang et al.
10,289,469	B2	5/2019	Fortino et al.
10,372,136	B2	8/2019	Yang et al.
10,489,972	B2	11/2019	Atsmon
10,580,158	B1	3/2020	Mousavian et al.
10,625,748	B1	4/2020	Dong et al.
10,635,110	B2	4/2020	Shashua et al.
10,635,844	B1*	4/2020	Roose G06F 30/20
10,730,517	B2	8/2020	Park et al.
10,739,778	B2	8/2020	Winkler et al.
10,816,978	B1*	10/2020	Schwalb G08G 1/012
10,942,030	B2	3/2021	Haque et al.
11,377,115	B2*	7/2022	Pretsch G06F 30/20
11,436,484	B2	9/2022	Farabet et al.
2004/0252864	A1	12/2004	Chang et al.
2007/0154068	A1	7/2007	Stein et al.
2007/0182528	A1	8/2007	Breed et al.
2009/0125177	A1	5/2009	Tanaka et al.
2009/0256840	A1	10/2009	Varadhan et al.
2013/0227569	A1	8/2013	Kohli
2014/0025338	A1	1/2014	Blount et al.
2015/0054824	A1	2/2015	Jiang
2016/0247290	A1	8/2016	Liu et al.
2016/0321074	A1	11/2016	Hung et al.
2017/0010108	A1	1/2017	Shashua
2017/0090478	A1	3/2017	Blayvas et al.
2017/0220876	A1	8/2017	Gao et al.
2017/0236013	A1	8/2017	Clayton et al.
2017/0259801	A1	9/2017	Abou-Nasr et al.
2017/0344808	A1	11/2017	El-Khamy et al.
2017/0364083	A1	12/2017	Yang et al.
2017/0371340	A1	12/2017	Cohen et al.

2018/0089833	A1	3/2018	Lewis et al.
2018/0121273	A1	5/2018	Fortino et al.
2018/0136332	A1	5/2018	Barfield, Jr. et al.
2018/0188059	A1	7/2018	Wheeler et al.
2018/0232663	A1	8/2018	Ross et al.
2018/0253095	A1*	9/2018	Ryan G05D 1/0088
2018/0267558	A1	9/2018	Tiwari et al.
2018/0276278	A1	9/2018	Cagan et al.
2018/0348374	A1	12/2018	Laddha et al.
2018/0349746	A1	12/2018	Vellespi-Gonzalez
2018/0370540	A1	12/2018	Yousuf et al.
2018/0373980	A1	12/2018	Huval
2019/0016285	A1	1/2019	Freienstein et al.
2019/0066328	A1	2/2019	Kwant et al.
2019/0071101	A1	3/2019	Emura et al.
2019/0072968	A1*	3/2019	Will, IV G08G 1/0129
2019/0102646	A1	4/2019	Redmon et al.
2019/0102668	A1*	4/2019	Yao G06N 3/04
2019/0147600	A1	5/2019	Karasev et al.
2019/0147610	A1	5/2019	Frossard et al.
2019/0171912	A1	6/2019	Vellespi-Gonzalez et al.
2019/0179979	A1*	6/2019	Melick G01P 21/02
2019/0212749	A1	7/2019	Chen et al.
2019/0213481	A1	7/2019	Godard et al.
2019/0250622	A1	8/2019	Nister et al.
2019/0258251	A1	8/2019	Ditty et al.
2020/0013176	A1	1/2020	Kang et al.
2020/0143205	A1	5/2020	Yao et al.

FOREIGN PATENT DOCUMENTS

DE	102015226762 A1	6/2017
EP	1930863 A2	6/2008
EP	2384009 A2	11/2011
KR	20120009590 A1	2/2012
WO	2012011713 A2	1/2012
WO	2016183074 A1	11/2016
WO	2018002910 A1	1/2018
WO	2018102717 A1	6/2018

OTHER PUBLICATIONS

Bojarski, Mariusz; "End to End Learning for Self-Driving Cars", https://arxiv.org/abs/1604.07316; Apr. 25, 2016, 9 pgs.

Final Office Action dated Dec. 17, 2021 in U.S. Appl. No. 16/355,328, 17 pgs.

Notice of Allowance dated Mar. 4, 2022 in U.S. Appl. No. 16/355,328, 5 pgs.

Asvadi, A., et al., "DepthCN: Vehicle Detection Using 3D-LIDAR and ConvNet", International Conference on Intelligent Transportation Systems (ITSC), IEEE, pp. 1-6 (Oct. 16, 2017), XP033330533.

Bidlack, C., et al., "Visual Robot Navigation using Flat Earth Obstacle Projection", Proceedings of the IEEE International Conference on Robotics and Automation, pp. 3374-3381 (May 8-13, 1994), XP010097532.

He, L., et al., "Learning Depth from Single Images with Deep Neural Network Embedding Focal Length", Arxiv.Org, Cornell University Library, pp. 1-14 (Mar. 27, 2018).

Kim, W., S., et al., "Depth Map Coding with Distortion Estimation of Rendered View", Proceedings of Spie, vol. 7543, pp. 7430B1-75430B10, (Jan. 17, 2010), XP055272237.

Stein, G.P., et al., "Vision-Based ACC with a Single Camera: Bounds on Range and Range Rate Accuracy", Proceedings of IEEE Intelligent Vehicle Symposium, pp. 1-6 (2003).

Suorsa, R., E., and Sridhar, B., "A Parallel Implementation of a Multisensor Feature-Based Range-Estimation Method", IEEE Transactions on Robotics and Automation, vol. 10, No. 6, pp. 1-34 (Dec. 1, 1994), XP000482433.

Tateno, K., et al., "CNN-SLAM: Real-time dense monocular SLAM with learned depth prediction", Arxiv.Org, Cornell University Library, pp. 6243-6252 (Apr. 11, 2017).

International Preliminary Report on Patentability received for PCT Application No. PCT/US2019/061820, mailed on May 27, 2021, 9 pgs.

(56) References Cited

OTHER PUBLICATIONS

International Preliminary Report on Patentability received for PCT Application No. PCT/US2019/068766, mailed on Jul. 8, 2021, 10 pgs.

International Preliminary Report on Patentability received for PCT Application No. PCT/US2019/068764, mailed on Jul. 8, 2021, 12 pgs.

Notice of Allowance dated Aug. 19, 2021 in U.S. Appl. No. 16/277,895, 10 pgs.

Notice of Allowance dated Feb. 17, 2021 in U.S. Appl. No. 16/241,005, 7 pages.

First Action Interview Office Action dated Mar. 1, 2021 in U.S. Appl. No. 16/355,328, 4 pages.

Notice of Allowance dated Mar. 26, 2021 in U.S. Appl. No. 16/265,780, 6 pages.

Final Office Action dated Apr. 15, 2021 in U.S. Appl. No. 16/277,895, 20 pages.

Non-Final Office Action dated Aug. 7, 2020 in U.S. Appl. No. 16/241,005, 22 pgs.

Non-Final Office Action dated Nov. 25, 2020 in U.S. Appl. No. 16/356,439, 22 pgs.

Notice of Allowance dated Jan. 19, 2021 in U.S. Appl. No. 16/286,329, 8 pgs.

Preinterview First Office Action dated Jan. 26, 2021 in U.S. Appl. No. 16/355,328, 5 pgs.

First Office Action dated Feb. 1, 2021 in U.S. Appl. No. 16/265,780, 15 Pgs.

International Preliminary Report on Patentability received for PCT Application No. PCT/US2019/016418, mailed on Aug. 13, 2020, 8 pgs.

International Preliminary Report on Patentability received for PCT Application No. PCT/US2019/018348, mailed on Aug. 27, 2020, 16 pgs.

International Preliminary Report on Patentability received for PCT Application No. PCT/US2019/019656, mailed on Sep. 3, 2020, 11 pgs.

International Preliminary Report on Patentability received for PCT Application No. PCT/US2019/022753, mailed on Oct. 1, 2020, 14 pgs.

International Preliminary Report on Patentability received for PCT Application No. PCT/US2019/024400, mailed on Oct. 8, 2020, 10 pgs.

Preinterview First Office Action dated Oct. 13, 2020 in U.S. Appl. No. 16/265,780, 16 pgs.

Non-Final Office Action dated Oct. 21, 2020 in U.S. Appl. No. 16/277,895, 13 pgs.

International Preliminary Report on Patentability International Patent Application No. PCT/US2019/012535 mailed Jul. 7, 2020.

Garnett, N., et al., (2017). "Real-time category-based and general obstacle detection for autonomous driving". In Proceedings of the IEEE International Conference on Computer Vision Workshops (pp. 198-205).

Non-Final Office Action mailed Jun. 24, 2020 in U.S. Appl. No. 16/286,329.

International Search Report and Written Opinion mailed Apr. 15, 2020 in Application No. PCT/US2019/061820 filed Nov. 15, 2019.

Zhong, Yiran et al.; "Self-Supervised Learning for Stereo Matching with Self-Improving Ability", Arxiv.org, Cornell University Library, 201 Olin Library Cornell University Ithaca, NY 14853, Sep. 4, 2017. 13 pgs.

Pang, Jiahao et al.; "Cascade Residual Learning: A Two-Stage Convolutional Neural Network for Stereo Matching", 2017 IEEE International Conference on Computer Vision Workshops (ICCVW), IEEE, Oct. 22, 2017, pp. 878-886. 9 pgs.

Godard, Clement et al.: "Unsupervised Monocular Depth Estimation with Left-Right Consistency", IEEE Computer Society Conference on Computer Vision and Pattern Recognition. Proceedings, IEEE Computer Society, US, Jul. 21, 2017. pp. 6602-6611. 10 pgs.

International Search Report and Written Opinion mailed Nov. 7, 2019 in International Patent Application No. PCT/US2019/022753, 22 pgs.

Kendall, Alex et al., "End-to-End Learning of Geometry and Context for Deep Stereo Regression" ARXIV.org, Cornell University Library, 201 Olin Library Cornell University Ithaca, NY 14853, Mar. 13, 2017, 10 pgs.

International Search Report and Written Opinion mailed Oct. 17, 2019 in International Patent Application No. PCT/US2019/012535, 24 pgs.

International Search Report and Written Opinion mailed Jul. 25, 2019 in International Patent Application No. PCT/US2019/018348, 22 pgs.

Andrew Tao et al: "Detectnet: Deep neural network for object detection in digits", NVIDIA Developer Blog, Aug. 11, 2015 (Aug. 11, 2016), OP055586923, Retrieved from the Internet:URL: https://devblogs.nvidia.com/detectnet-deep-neural-network-object-detection-digits/ [retrieved on May 8, 2019].

Rothe, Rasmus; Gulliaumin, Matthieu; Van Gool, Luc: "Non-maximum Suppression for Object Detection by Passing Messages Between Windows"; "1" In: D. Cremers et al.(Eds.): "Computer Vision—ACCV 2014", 2015, Springer International Publishing, Cham, Switzerland, WP0473211944, ISBN: 978-3-319-16865-4, vol. LNCS900, pp. 290-306, DOI:10.1007/978-3-319-16865-4-19.

Weber Michael et al: "DeepTLR: A single deep convolutional network for detection and classification of traffic lights", 2016 IEEE Intelligent Vehicles Symposium (Iv). IEEE, Jun. 19, 2016 (Jun. 19, 2016), pp. 342-348, XP032938986, DOI: 10.1109/IVS.2016.7535408.

Bach Martin et al: "Multi-camera traffic light recognition using a classifying Labeled Multi-Bernoulli filter", 2017 IEEE Intelligent Vehicles Symposium (IV), IEEE, Jun. 11, 2017 (Jun. 11, 2017), pp. 1045-1051, KP033133881, DOI: 10.1109/IVS.2017.7995852.

Liu Hau et al: "Neural Person Search Machines", 2017 IEEE International Conference on Computer Vision (ICCV), IEEE, Oct. 22, 2017 (Oct. 2, 2017), pp. 493-501, XP033282904, DOI: 10.1109/ICCV.2017.61.

Arvind Jayarahan et al: "Creating 3D Virtual Driving Environments for Simulation-Aided Development of Autonomous Driving and Active Safety", SAE Technical Paper Series, vol. 1, Mar. 28, 2017 (Mar. 28, 2017), KP055518353, US, ISSN: 0148-7191, DOI: 10.4271/2017-01-0107, p. 1-p. 4.

International Search Report and Written Opinion mailed Jun. 26, 2019 in International Patent Application No. PCT/US2019/024400, 15 pgs.

International Search Report and Written Opinion mailed Aug. 26, 2019 in International Patent Application No. PCT/US2019/022592, 18 pgs.

"Systems and Methods for Safe and Reliable Autonomous Vehicles", U.S. Appl. No. 62/584,549, filed Nov. 10, 2017.

"System and Method for Controlling Autonomous Vehicles", U.S. Appl. No. 62/614,466, filed Jan. 7, 2018.

"System and Method for Safe Operation of Autonomous Vehicles", U.S. Appl. No. 62/625,351, filed Feb. 2, 2018.

"Conservative Control for Zone Driving of Autonomous Vehicles Using Safe Time of Arrival", U.S. Appl. No. 62/628,831, filed Feb. 9, 2018.

"System and Method for Sharing Camera Data Between Primary and Backup Controllers in Autonomous Vehicle Systems", U.S. Appl. No. 62/629,822, filed Feb. 13, 2018.

"Pruning Convolutional Neural Networks for Autonomous Vehicles and Robotics", U.S. Appl. No. 62/630,445, filed Feb. 14, 2018.

"Methods for accurate real-time object detection and for determining confidence of object detection suitable for autonomous vehicles", U.S. Appl. No. 62/631,781, filed Feb. 18, 2018.

"System and Method for Autonomous Shuttles, Robo-Taxis, Ride-Sharing and On-Demand Vehicles", U.S. Appl. No. 62/635,503, filed Feb. 26, 2018.

"Convolutional Neural Networks to Detect Drivable Freespace for Autonomous Vehicles", U.S. Appl. No. 62/643,665, filed Mar. 15, 2018.

(56) **References Cited**

OTHER PUBLICATIONS

"Deep Neural Network for Estimating Depth from Stereo Using Semi-Supervised Learning", U.S. Appl. No. 62/646,148, filed Mar. 21, 2018.

"Video Prediction Using Spatially Displaced Convolution", U.S. Appl. No. 62/646,309, filed Mar. 21, 2018.

"Video Prediction Using Spatially Displaced Convolution", U.S. Appl. No. 62/647,545, filed Mar. 23, 2018.

"Geometric Shadow Filter for Denoising Ray-Traced Shadows", U.S. Appl. No. 62/644,385, filed Mar. 17, 2018.

"Energy Based Reflection Filter for Denoising Ray-Traced Glossy Reflections", U.S. Appl. No. 62/644,386, filed Mar. 17, 2018.

"Distance Based Ambient Occlusion Filter for Denoising Ambient Occlusions", U.S. Appl. No. 62/644,601, filed Mar. 19, 2018.

"Adaptive Occlusion Sampling of Rectangular Area Lights with Voxel Cone Tracing", U.S. Appl. No. 62/644,806, filed Mar. 19, 2018.

"System and Methods for Advanced AI-Assisted Vehicles", U.S. Appl. No. 62/648,358, filed Mar. 26, 2018.

"Network Injection Rate Limiting", U.S. Appl. No. 62/648,326, filed Mar. 26, 2018.

"Network Synchronization Using Posted Operation Tracking For Flush Semantics", U.S. Appl. No. 62/648,333, filed Mar. 26, 2018.

First Office Action for Chinese Patent Application No. 2019800225114, filed Sep. 25, 2020, mailed May 11, 2023, 10 pgs.

Jayaraman, et al.; "Creating 3D Virtual Driving Environments for Simulation-Aided Development of Autonomous Driving and Active Safety", SAE Technical Paper Series, Jul. 20, 2015, 6 pgs.

Farabet, et al.; Second Office Action for Chinese Patent Application No. 201980022511.4, filed Sep. 25, 2020, mailed Jul. 7, 2023, 11 pgs.

Farabet, Clement; Third Office Action for Chinese Patent Application No. 201980022511.4, Sep. 25, 2020, mailed Oct. 10, 2023, 4 pgs.
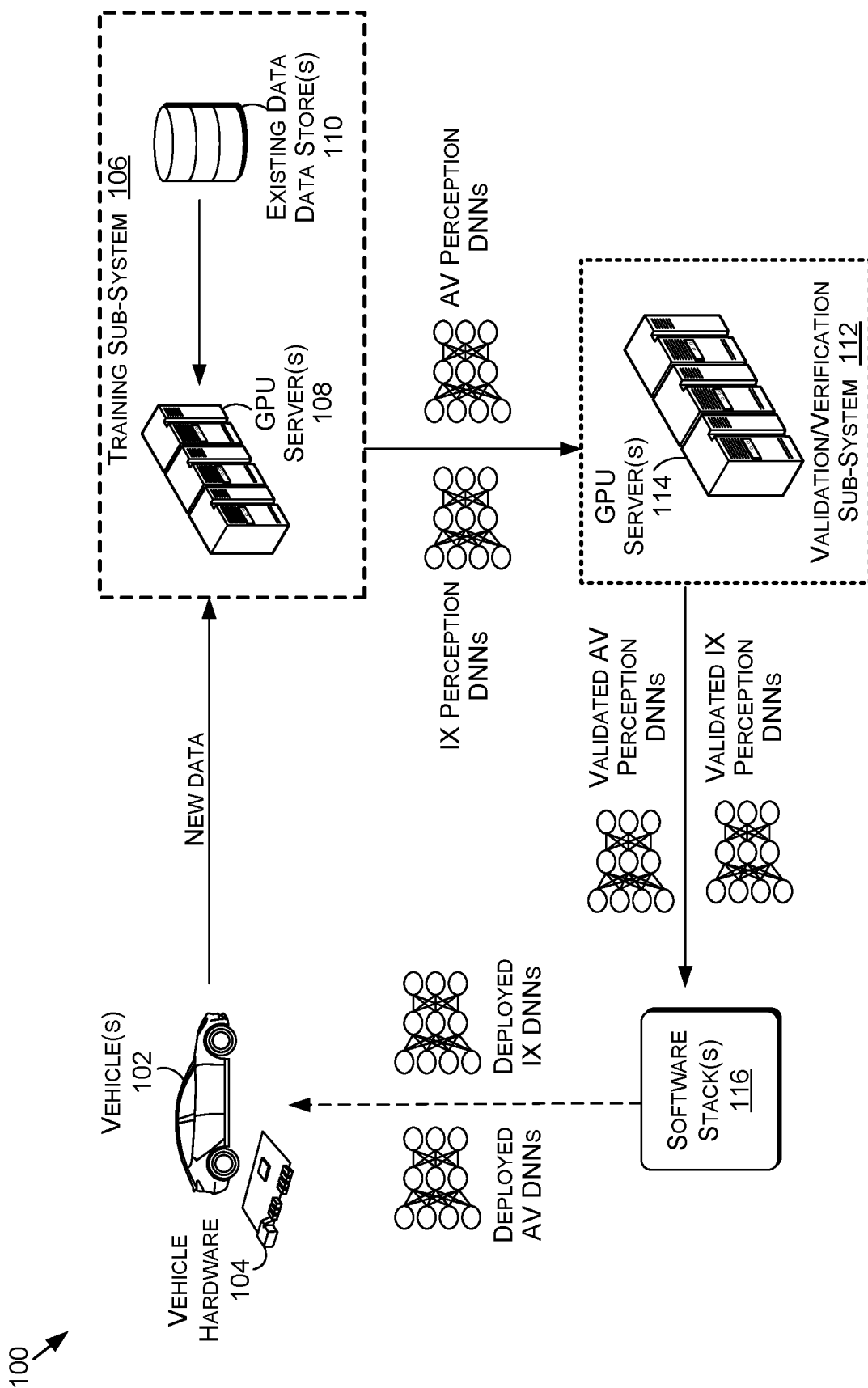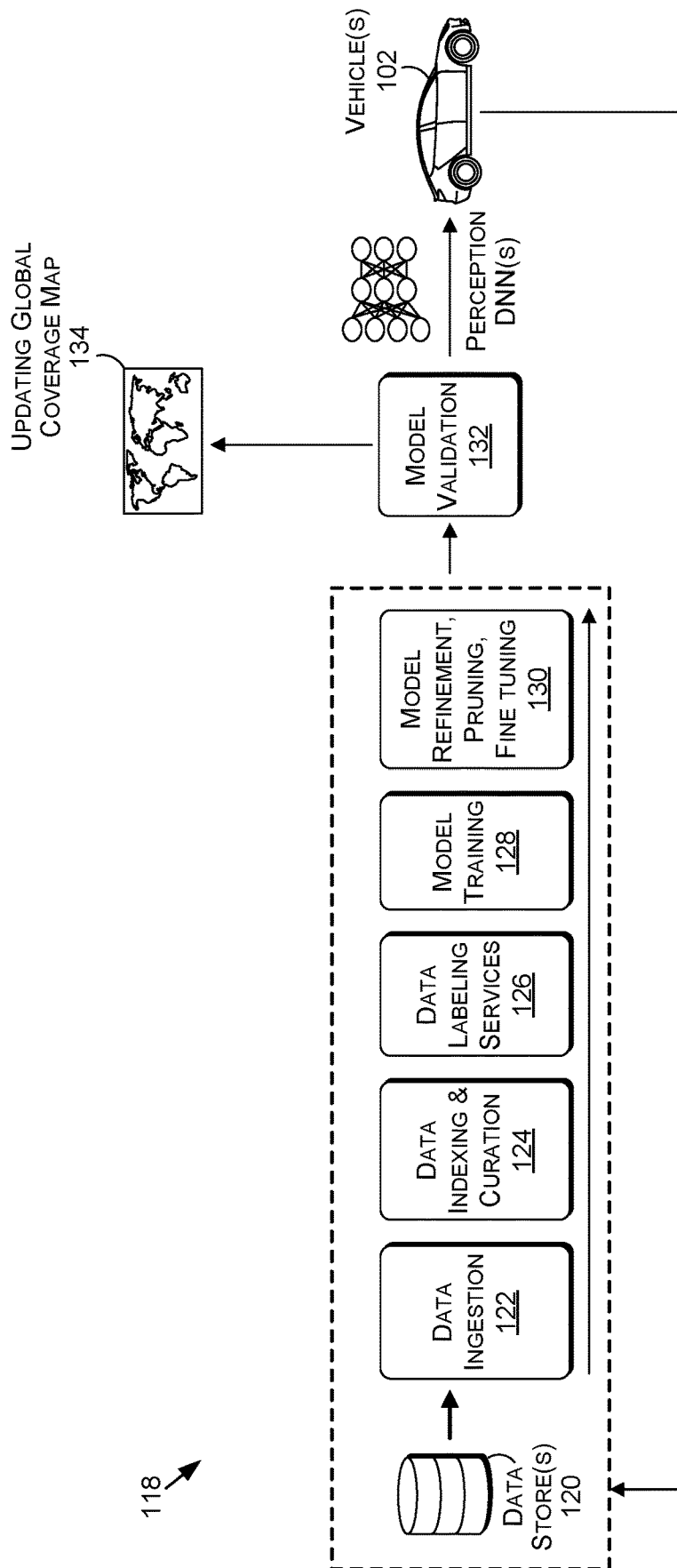
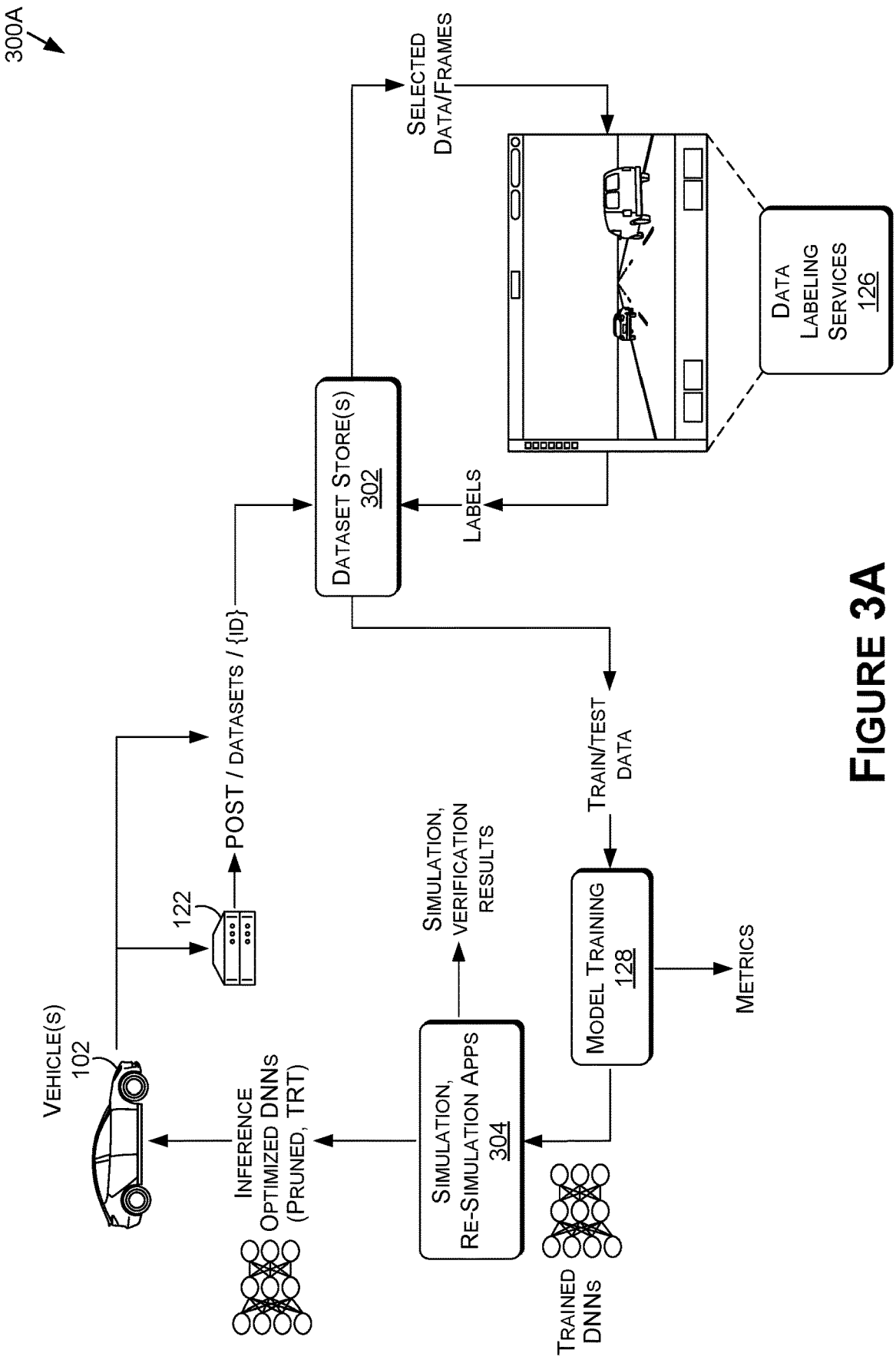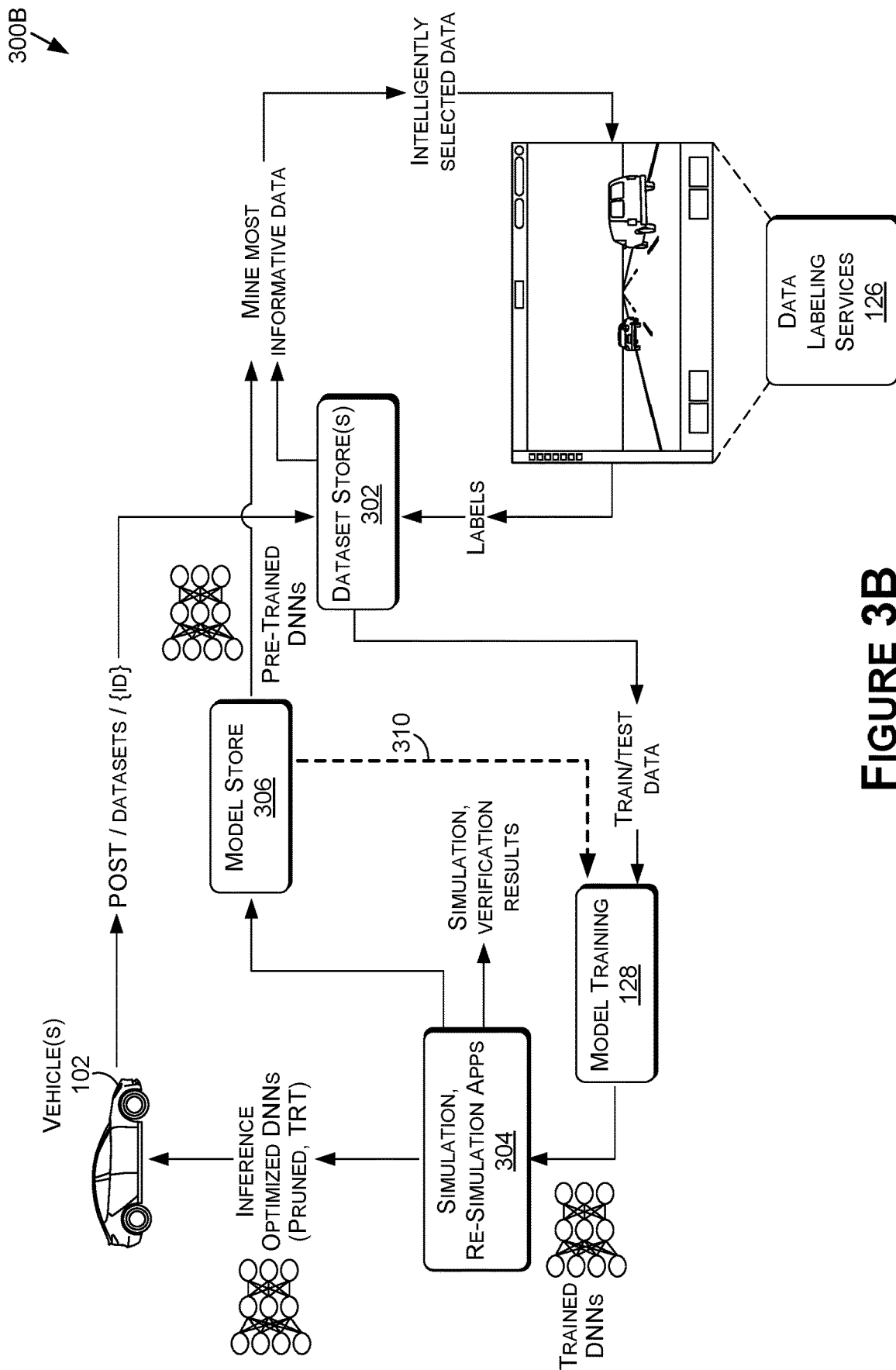* cited by examiner

FIGURE 1

FIGURE 2

300A

DATASET STORE(S) 302

SELECTED DATA/FRAMES

DATA LABELING SERVICES 126

LABELS

POST / DATASETS / {ID}

TRAIN/TEST DATA

VEHICLE(S) 102

122

INFERENCE OPTIMIZED DNNS (PRUNED, TRT)

SIMULATION, VERIFICATION RESULTS

SIMULATION, RE-SIMULATION APPS 304

MODEL TRAINING 128

METRICS

TRAINED DNNS

FIGURE 3A

300B

INTELLIGENTLY SELECTED DATA

MINE MOST INFORMATIVE DATA

DATA LABELING SERVICES 126

DATASET STORE(S) 302

LABELS

PRE-TRAINED DNNs

POST / DATASETS / {ID}

MODEL STORE 306

310

TRAIN/TEST DATA

SIMULATION, VERIFICATION RESULTS

MODEL TRAINING 128

VEHICLE(S) 102

INFERENCE OPTIMIZED DNNs (PRUNED, TRT)

SIMULATION, RE-SIMULATION APPS 304

TRAINED DNNs

FIGURE 3B

FIGURE 3C

FIGURE 3D

SENSOR SWITCH(ES) 408

VEHICLE HARDWARE 104

GPU(s) 404

VEHICLE SIMULATOR COMPONENT(S) 406

SIMULATOR COMPONENT(S) 402

412B

418

414

412A

416

410

400A

FIGURE 4A

400B

VEHICLE HARDWARE 104

VEHICLE SIMULATOR COMPONENT(S) (HIL) 406

VEHICLE SIMULATOR COMPONENT(S) (SIL) 420
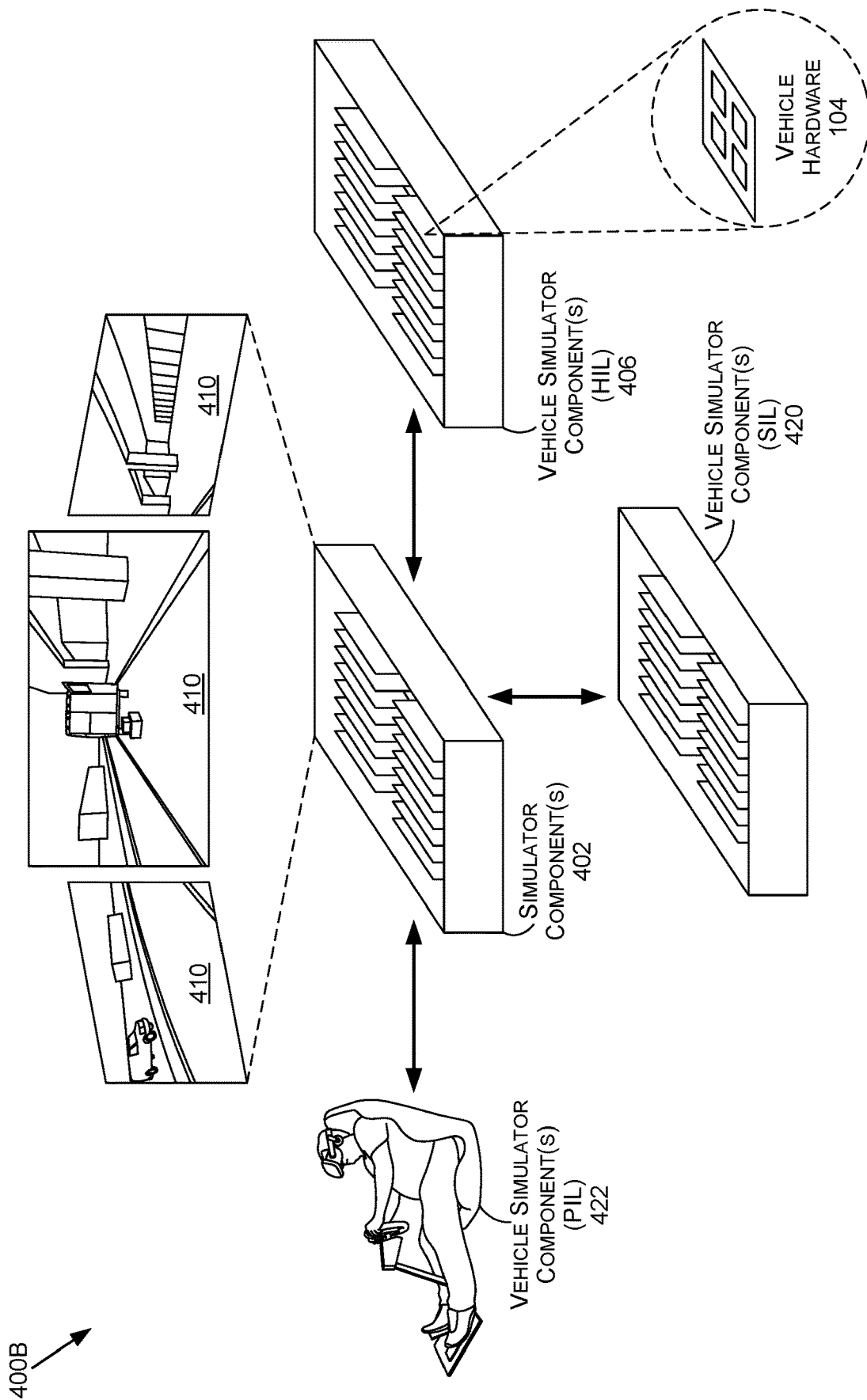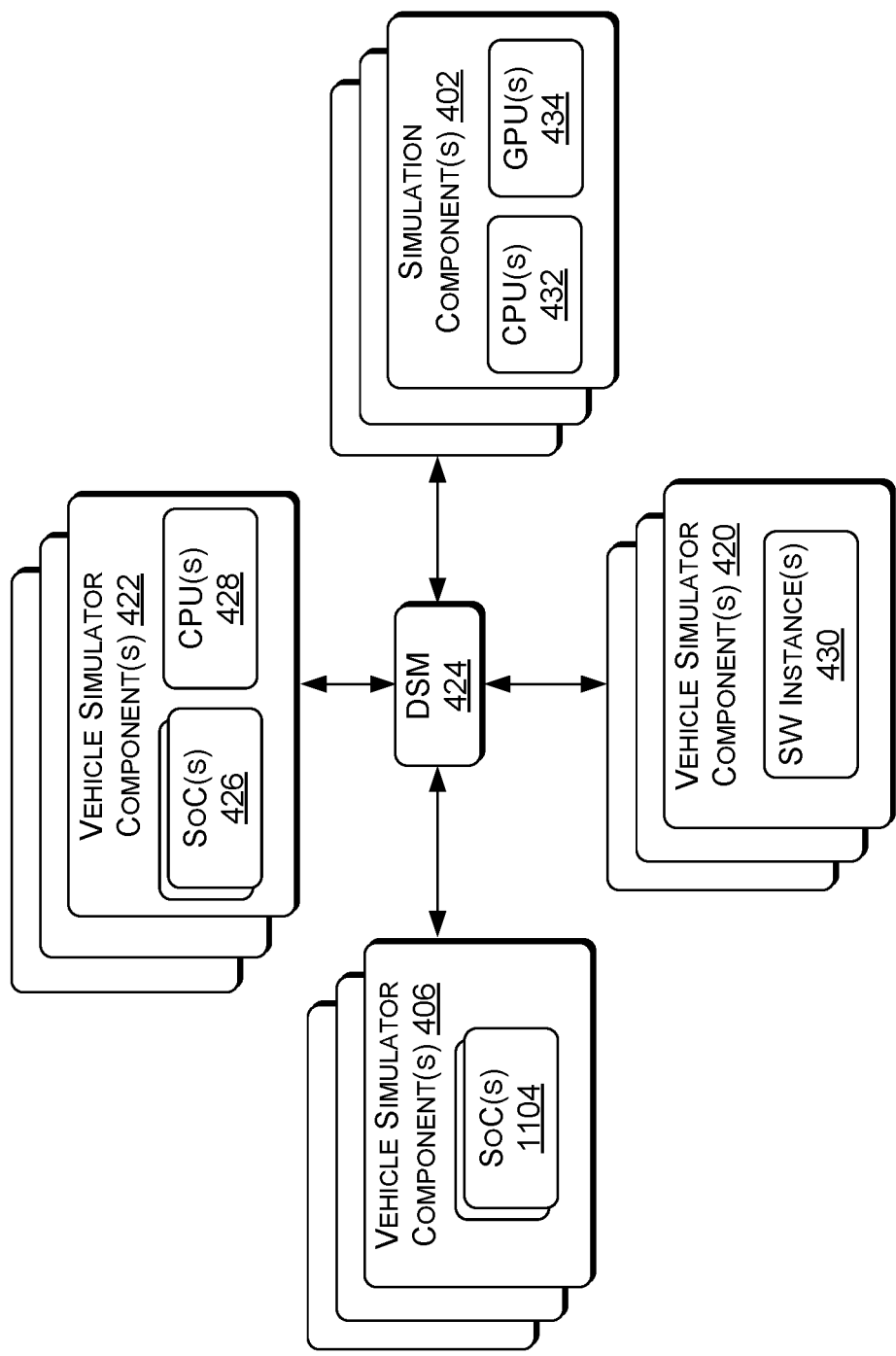
410

410

410

SIMULATOR COMPONENT(S) 402

VEHICLE SIMULATOR COMPONENT(S) (PIL) 422

FIGURE 4B

FIGURE 4C

FIGURE 4D

FIGURE 4F

FIGURE 4E

500

TRANSMIT, FROM A FIRST HARDWARE COMPONENT TO A
SECOND HARDWARE COMPONENT, SIMULATION DATA
B502

RECEIVE, BY THE FIRST HARDWARE COMPONENT AND
FROM THE SECOND HARDWARE COMPONENT, A SIGNAL
B504

UPDATE, BY THE FIRST HARDWARE COMPONENT, ONE OR
MORE ATTRIBUTES OF A VIRTUAL OBJECT WITHIN A
SIMULATED ENVIRONMENT
B506

FIGURE 5

FIGURE 6A

600B

VEHICLE HARDWARE 104

VEHICLE HARDWARE 104

626

624

VIRTUAL SENSOR/CODEC 618(1)

VIRTUAL SENSOR/CODEC 618(2)

· · ·

VIRTUAL SENSOR/CODEC 618(N)

VIRTUAL SENSOR/CODEC 620(1)

VIRTUAL SENSOR/CODEC 620(2)

· · ·

VIRTUAL SENSOR/CODEC 620(N)

AI UNIVERSE 622

GI ENGINE 634

ASSET DATA STORE(S) 636

SIMULATION ENGINE 630

AI ENGINE 632

628

FIGURE 6B

700

SOFTWARE STACK(S)
706

SENSOR DATA STREAMS
FOR CURRENT TIME SLICE

DRIVING COMMANDS AND
NEW SENSOR STATE

CODEC(S)
704

CURRENT STATE
AND SENSOR DATA

DRIVING COMMANDS AND
NEW SENSOR STATE

KPI FRAMEWORK
710

EVALUATION
RESPONSE

SIM/RESIM
702

UPDATED
OBJECT
STATE

EGO-OBJECT
DYNAMICS
708

CURRENT
STATE

DRIVING
COMMANDS

FIGURE 7

REPORT

KPI EVALUATION COMPONENT
802

SOFTWARE STACK(S)
116

RE-SIMULATOR/SIMULATOR
804

LOGS
806

FIGURE 8

900

RECEIVE SIMULATION DATA REPRESENTATIVE OF A SIMULATED ENVIRONMENT OF A SIMULATION HOST DEVICE
B902

GENERATE VIRTUAL SENSOR DATA FOR EACH OF A DYNAMICALLY CONFIGURABLE NUMBER OF VIRTUAL SENSORS
B904

ENCODE THE VIRTUAL SENSOR DATA
B906

COMPUTE, BY ONE OR MORE MACHINE LEARNING MODELS, AT LEAST ONE OUTPUT
B908

TRANSMIT, TO THE SIMULATION HOST DEVICE, A SIGNAL
B910

FIGURE 9

1000

RECEIVE PHYSICAL SENSOR DATA GENERATED BY A
PHYSICAL SENSOR
B1002

TRAIN A MACHINE LEARNING MODEL (MLM) USING THE
PHYSICAL SENSOR DATA
B1004

RECEIVE VIRTUAL SENSOR DATA GENERATED BY A VIRTUAL
SENSOR
B1006

APPLY THE VIRTUAL SENSOR DATA TO THE TRAINED MLM
B1008

COMPUTE, BY THE TRAINED MLM, AN OUTPUT
B1010

CONTROL A VIRTUAL OBJECT WITHIN A SIMULATED
ENVIRONMENT BASED AT LEAST IN PART ON THE OUTPUT
B1012

FIGURE 10

FIGURE 11A

FIGURE 11B

STEREO CAMERA 1168

MID-RANGE CAMERA, WING MIRROR MOUNT 1198

SURROUND CAMERA 1174

MID-RANGE CAMERA, WING MIRROR MOUNT 1198

SURROUND CAMERA(S) 1174

SURROUND CAMERA(S) 1174

INFRARED CAMERA 1172

WIDE VIEW CAMERA 1170

LONG-RANGE CAMERA 1198

LONG-RANGE CAMERA 1198

STEREO CAMERA 1168

102

| GNSS SENSOR(S) 1158 | RADAR SENSOR(S) 1160 | ULTRASONIC SENSOR(S) 1162 | LIDAR SENSOR(S) 1164 | IMU SENSOR(S) 1166 | MICROPHONE(S) 1196 |

| STEREO CAM(S) 1168 | WIDE-VIEW CAM(S) 1170 | INFRARED CAM(S) 1172 | SURROUND CAM(S) 1174 | LONG/MID-RANGE CAM(S) 1198 |

INFOTAINMENT SoC 1130

INSTRUMENT CLUSTER 1132

HMI DISPLAY 1134

ADAS SYSTEM 1138

CONTROLLER(S) 1136

SoC 1104(B)

SoC 1104(A)

CPU(S) 1106   GPU(S) 1108

PROCESSOR(S) 1110

CACHE(S) 1111

ACCELERATOR(S) 1114

DATA STORE(S) 1116

CPU(S) 1118

GPU(S) 1120   1102

HD MAP 1122

1126

NETWORK INTERFACE 1124

DATA STORE(S) 1128

| STEERING SENSOR(S) 1140 | VIBRATION SENSOR(S) 1142 | SPEED SENSOR(S) 1144 | BRAKE SENSOR SYSTEM 1146 | PROPULSION SYSTEM 1150 | STEERING SYSTEM 1154 |

BRAKE ACTUATORS 1148

THROTTLE/ ACCELERATOR 1152

STEERING ACTUATORS 1156

102

FIGURE 11C

FIGURE 11D

1200

MEMORY
1204

I/O PORT(S)
1212

CPU(S)
1206

I/O COMPONENTS
1214

GPU(S)
1208

POWER SUPPLY
1216

COMM. INTERFACE
1210

PRESENTATION
COMPONENT(S)
1218

1202

FIGURE 12

TRAINING, TESTING, AND VERIFYING AUTONOMOUS MACHINES USING SIMULATED ENVIRONMENTS

CROSS-REFERENCE TO RELATED APPLICATIONS

This application is a continuation of U.S. application Ser. No. 16/366,875, filed Mar. 27, 2019, which claims the benefit of U.S. Provisional Application No. 62/648,399, filed on Mar. 27, 2018, which is hereby incorporated by reference in its entirety.

This application is related to U.S. Non-Provisional application Ser. No. 16/356,439, filed on Mar. 18, 2019, U.S. Non-Provisional application Ser. No. 16/355,328, filed on Mar. 15, 2019, U.S. Non-Provisional application Ser. No. 16/286,329, filed on Feb. 26, 2019, U.S. Non-Provisional application Ser. No. 16/277,895, filed on Feb. 15, 2019, and U.S. Non-Provisional application Ser. No. 16/265,780, filed on Feb. 1, 2019, U.S. Non-Provisional application Ser. No. 16/241,005, filed on Jan. 7, 2019, U.S. Non-Provisional application Ser. No. 16/186,473, filed on Nov. 9, 2018, each of which is hereby incorporated by reference in its entirety.

BACKGROUND

Autonomous vehicles and semi-autonomous vehicles rely on machine learning, and specifically deep neural networks (DNNs), for performing any number of operations for operating, piloting, and navigating the vehicle. For example, DNNs may be used for object detection, lane and road boundary detection, safety analysis, drivable free-space analysis, control generation during vehicle maneuvers, and the like. However, in order to verify and validate the usage of the DNNs in autonomous or semi-autonomous vehicles, the DNNs need to be trained on a large amount of data, which requires vast amounts of time and effort, and yet still does not always guarantee universally accurate or usable results.

For example, conventional systems often rely on data generated by physical vehicles navigating real-world environments to train the DNNs prior to deployment in working models. This approach has several limitations, however. For example, vehicles can only navigate to so many places, recreating dangerous or unique situations is difficult in the real-world environment, and testing the DNNs in these real-world environments may be dangerous. For example, especially where a DNN is used for obstacle avoidance or other safety measures, testing the DNNs in real-world environments may not be practical. On the other hand, an automaker likely will not deploy an autonomous vehicle into the real-world until an acceptable level of safety has been achieved. As a result, these competing interests make generating a sound, safe, accurate, and reliable autonomous driving system increasingly difficult.

SUMMARY

Embodiments of the present disclosure relate training, testing, and verifying autonomous machines using simulated environments. Systems and methods are disclosed for training, testing, and/or verifying one or more features of a real-world system—such as a software stack for use in autonomous vehicles and/or robots.

In contrast to conventional systems, such as those described above, the systems of the present disclosure leverage a simulated environment to test one or more

autonomous driving software stacks that include a multitude of DNNs. For example, physical sensor data, virtual sensor data, or a combination thereof may be used to train the DNNs of the software stack(s). Once trained, the DNNs may be tested, verified, and validated within a simulation system that generates a simulated environment for controlling a virtual object using the software stack(s). Simulation data from the simulated environment may be input into the DNNs of the software stack(s), and the DNNs may generate outputs. In some examples, the outputs may be used to control the virtual object within the simulated environment, to determine how the virtual object (and thus a physical object that corresponds to the virtual object) may perform in any number of different situations. In other examples, the outputs may be used to test the accuracy of the DNNs, and the results may be used to generate more data for further training (e.g., data that the DNNs are least consistent processing accurately), fine-tune the DNNs, verify the DNNs, and/or validate the DNNs. In any example, the simulated environment may be generated to create difficult to navigate, dangerous, unsafe, and/or otherwise unpredictable situations for the virtual object to navigate. As a result, previously untested scenarios (e.g., due to safety concerns, difficulty of reproduction, etc.) may be tested, repeated, and improved upon within the simulated environment.

In some examples, vehicle hardware configured for installation within an autonomous vehicle may be used to execute the software stack(s) within the simulated environment. In addition, the virtual sensor data may be encoded to a format that is familiar to the software stack(s) (e.g., is bit-to-bit the same as the physical sensor data used for training the DNNs). As a result, the testing, training, verification, and/or validation of the DNNs may be substantially identical to employing the hardware/software components in a physical vehicle in a real-world environment.

BRIEF DESCRIPTION OF THE DRAWINGS

The present systems and methods for training, testing, and verifying autonomous machines using simulated environments is described in detail below with reference to the attached drawing figures, wherein:

FIG. 1 is an example system for re-simulation, in accordance with some embodiments of the present disclosure;

FIG. 2 includes a data flow diagram for a process of testing, training, verifying, and/or validating neural networks, in accordance with some embodiments of the present disclosure;

FIGS. 3A-3D include workflows used for training DNNs, in accordance with some embodiments of the present disclosure;

FIGS. 4A-4F are example illustrations of a simulation system, in accordance with some embodiments of the present disclosure;

FIG. 5 is a flow diagram showing a method for generating a simulated environment using a hardware-in-the-loop object, in accordance with some embodiments of the present disclosure;

FIG. 6A is an example illustration of a simulation system at runtime, in accordance with some embodiments of the present disclosure;

FIG. 6B includes a cloud-based architecture for a simulation system, in accordance with some embodiment of the present disclosure;

FIG. 7 includes a data flow diagram illustrating a process 700 for re-simulation or simulation using one or more codecs, in accordance with some embodiments of the present disclosure;

FIG. 8 includes a data flow diagram for key performance indicator (KPI) analysis and observation, in accordance with some embodiments of the present disclosure;

FIG. 9 is a flow diagram showing a method for controlling a virtual object in a simulated environment, in accordance with some embodiments of the present disclosure;

FIG. 10 is a flow diagram showing a method for controlling a virtual object in a simulated environment using machine learning models trained on physical sensor data, in accordance with some embodiments of the present disclosure;

FIG. 11A is an illustration of an example autonomous vehicle, in accordance with some embodiments of the present disclosure;

FIG. 11B is an example of camera locations and fields of view for the example autonomous vehicle of FIG. 11A, in accordance with some embodiments of the present disclosure;

FIG. 11C is a block diagram of an example system architecture for the example autonomous vehicle of FIG. 11A, in accordance with some embodiments of the present disclosure;

FIG. 11D is a system diagram for communication between cloud-based server(s) and the example autonomous vehicle of FIG. 11A, in accordance with some embodiments of the present disclosure; and

FIG. 12 is a block diagram of an example computing device suitable for use in implementing some embodiments of the present disclosure.

DETAILED DESCRIPTION

Systems and methods disclosed are related to training, testing, and verifying autonomous machines or objects in simulated environments. The present disclosure may be described generally with respect to an example autonomous or semi-autonomous vehicle 102 (alternatively referred to herein as "vehicle 102" or "autonomous vehicle 102"), an example of which is described in more detail herein with respect to FIGS. 11A-11D. However, this is not intended to be limiting. For example, and without departing from the scope of the present disclosure, the systems, methods, and/or processes described herein may be applicable to non-autonomous vehicles, robots, unmanned aerial vehicles, and/or any other type of vehicle or object.
Re-Simulation System

Now referring to FIG. 1, FIG. 1 is an example system 100 for re-simulation, in accordance with some embodiments of the present disclosure. For example, the system 100 may be used for training, testing, verifying, deploying, updating, re-verifying, and/or deploying one or more neural networks for use in an autonomous vehicle, a semi-autonomous vehicle, a robot, and/or another object. In some examples, the system 100 may include some or all of the component, features, and/or functionality of system 1176 of FIG. 11D, and/or may include additional and/or alternative components, features, and functionality of the system 1176. It should be understood that this and other arrangements described herein are set forth only as examples. Other arrangements and elements (e.g., machines, interfaces, functions, orders, groupings of functions, etc.) may be used in addition to or instead of those shown, and some elements may be omitted altogether. Further, many of the elements

described herein are functional entities that may be implemented as discrete or distributed components or in conjunction with other components, and in any suitable combination and location. Various functions described herein as being performed by entities may be carried out by hardware, firmware, and/or software. For instance, various functions may be carried out by a processor executing instructions stored in memory.

One or more vehicles 102 may collect sensor data from one or more sensors of the vehicle(s) 102 in real-world (e.g., physical) environments. The sensors of the vehicle(s) 102 may include, without limitation, global navigation satellite systems sensor(s) 1158 (e.g., Global Positioning System sensor(s)), RADAR sensor(s) 1160, ultrasonic sensor(s) 1162, LIDAR sensor(s) 1164, inertial measurement unit (IMU) sensor(s) 1166 (e.g., accelerometer(s), gyroscope(s), magnetic compass(es), magnetometer(s), etc.), microphone(s) 1196, stereo camera(s) 1168, wide-view camera(s) 1170 (e.g., fisheye cameras), infrared camera(s) 1172, surround camera(s) 1174 (e.g., 360 degree cameras), long-range and/or mid-range camera(s) 1198, speed sensor(s) 1144 (e.g., for measuring the speed of the vehicle 102), vibration sensor(s) 1142, steering sensor(s) 1140, brake sensor(s) (e.g., as part of the brake sensor system 1146), and/or other sensor types. The vehicle(s) 102 may include autonomous vehicles, semi-autonomous vehicles, non-autonomous vehicles, and/or may include objects other than vehicles 102, such as robots, drones, unmanned aerial vehicles (UAVs), etc.

The vehicle(s) 102 may include vehicle hardware 104. For example, the vehicle hardware 104 may be responsible for managing the sensor data generated by the sensors (e.g., using a sensor manager of an autonomous driving software stack being executed by the vehicle hardware 104). The autonomous driving software stack being executed using the vehicle hardware 104 may further include a world state manager that manages the world using one or more maps (e.g., 3D maps), localization component(s), perception component(s), and/or the like. In addition, the autonomous driving software stack may include planning component(s) (e.g., as part of a planning layer), control component(s) (e.g., as part of a control layer), actuation component(s) (e.g., as part of an actuation layer), obstacle avoidance component(s) (e.g., as part of an obstacle avoidance layer), and/or other component(s). In any example, the vehicle hardware 104 may include the hardware of the vehicle 102 that is used to control the vehicle 102 through real-world environments based on the sensor data, one or more machine learning models (e.g., neural networks), and/or the like. As such, the vehicle hardware 104 may be configured for installation within the vehicle 102 and for use by the vehicle 102 in executing an autonomous driving software stack for controlling, at least in part, the vehicle 102 through a real-world physical environment(s).

The sensor data collected by the sensors of the vehicle(s) 102, in addition to existing sensor data (e.g., sensor data stored in the data store(s) 110), may be used by a training sub-system 106. The training sub-system 106 may include a cloud-based deep learning infrastructure that may use artificial intelligence to analyze the sensor data received from the vehicle(s) 102 and/or stored in the data store(s) 110, and incorporate or train up-to-date, real-time neural networks (and/or other machine learning models) for real-time intelligent inferencing. In some examples, the training sub-system 106 may include one or more graphics processing unit (GPU) servers 108. For example, the training sub-system 106 may include a datacenter with GPUs, TPUs, CPUs, and/or other processor types. As such, the use of GPU

5

6

with respect to GPU server(s) **108** is not intended to be limiting, and in some examples, the GPU server(s) **108** may not include GPU(s).

The training sub-system **106** may train and/or test any number of machine learning models, including deep neural networks (DNNs), such as neural networks for performing operations associated with one or more layers of the autonomous driving software stack and/or an in-cabin experience (IX) software stack. For example, one or more autonomous vehicle (AV) perception DNNs may be trained and/or tested, where the AV perception DNNs may be used for detecting lanes and boundaries on driving surfaces, for detecting drivable free-space, for detecting traffic poles or signs, for detecting traffic lights, for detecting objects in the environment (e.g., vehicles, pedestrians, animals, inanimate objects, etc.), for detecting wait conditions and intersections, and/or the like. As another example, one or more in-cabin experience (IX) perception DNNs may be trained and/or tested, where the IX perception DNNs may be used for monitoring passengers and drivers within the vehicle **102**. For example, one or more IX perception DNNs may be trained to determine a state of the driver—such as, but not limited to, gaze tracking, head pose tracking, drowsiness detection, sleepiness, eye openness, emotion detection, heart rate monitoring, liveliness of the driver, driver impairment, and/or the like.

Once trained and/or tested, the IX perception DNNs, the AV perception DNNs, and/or other DNNs may be validated and/or verified by a validation/verification sub-system **112**. The validation/verification sub-system **112** may include similar components and/or features as the training sub-system **106**. In some examples, the training sub-system **106** and the validation/verification sub-system **112** may include the same hardware components, while in other examples the hardware components may differ. The validation/verification sub-system **112** may verify and/or validate performance, accuracy, and/or other criteria associated with the DNNs. Once verified and/or validated, the validated AV perception DNNs and/or the validated IX perception DNNs may be incorporated into software stack(s) **116** (e.g., the IX software stack and/or the autonomous driving software stack). Once incorporated into the software stack(s) **116**, the vehicle(s) **102** may execute the software stack(s) using the vehicle hardware **104** to control the vehicle(s) **102** within real-world environments. Although the vehicle(s) **102** in FIG. **1** are illustrated as collecting data and using the deployed DNNs, this is not intended to be limiting. For example, some vehicles **102** may be used for data capture, while other vehicles **102** may be used for autonomous driving applications using the deployed DNNs. As such, in some examples, first vehicles **102** may collect the data, and second vehicles **102** may use the software stack(s) **116** once the data collected from the first vehicles **102** has been used to train the deployed DNNs.

As such, the system **100** may include a re-simulation system that uses physical sensor data generated by vehicle(s) **102** in real-world environments to train, test, verify, and/or validate one or more DNNs for use in the software stack(s) **116**. In some examples, as described herein, the re-simulation system **100** may overlap with simulation system(s) **400A**, **400B**, **400C**, and/or **400D** in that at least some of the testing, training, verification, and/or validation may be performed within a simulated environment.

Now referring to FIG. **2**, FIG. **2** includes a data flow diagram for a process **118** of testing, training, verifying, and/or validating neural networks, in accordance with some embodiments of the present disclosure. The process **118** may include data ingestion of new driving data (e.g., sensor data) captured and/or generated by one or more vehicles **102** in real-world environments and/or simulated or virtual sensor data from one or more simulated environments. The process **118** may further include data indexing and curation **124**, data labeling services **126**, model training **128**, model refinement, pruning, and/or fine tuning **130**, model validation **132**, and/or updating global coverage maps **134**. The process **118** may include a training loop, whereby new data is generated by the vehicle(s) **102**, used to train, test, verify, and/or validate one or more perception DNNs, and the trained or deployed DNNs are then used by the vehicle(s) **102** to navigate real-world environments.

The data store(s) **120** may store sensor data and/or virtual sensor data generated by one or more real-world sensors of one or more vehicle(s) **102** and/or virtual sensors of one or more virtual vehicles, respectively.

Data ingestion **122** may include generating and/or recording the data output by the vehicle(s) **102** autonomous vehicle platform (e.g., the vehicle hardware **104** and/or the software stack(s) **116**). For a non-limiting example, the data may be written out to solid state drives (SSDs) and/or downloaded by wire and/or wirelessly to the data store(s) **120**.

Data indexing and curation **124** may include indexing metadata associated with the data output by the vehicle(s) **102** for further search and/or retrieval. Search indexes may be used to retrieve specific segments of the data, which may then be tagged and/or flagged for further processing. In some examples, raw data may be stored in a lossless format to allow for further pre-processing and/or quantization. In such examples, an on-demand transcoding service may transform the raw data into various target formats (e.g., MPEG, JPEG, FP16, etc.) and may feed or input the transformed data to one or more processing pipelines (e.g., labeling, DNN training, re-simulation, etc.). Exported datasets may be stored in a dataset store, which may be a service that handles immutable datasets for further processing. Once the datasets are stored, the datasets may be used and re-used to reproduce training results exactly, or run and re-run simulation jobs.

Data labeling services **126** may be used to tag and/or label the data, the raw data, the transformed data, and/or any other data used in the process **118**. In examples, the tagging and/or labeling may be performed by a human, by a machine, or by a combination thereof.

Model training **128** may use a deep-learning platform to define training applications and to run the training application on a compute cluster (e.g., using the training sub-system **106** of FIG. **1**). The compute cluster may include one or more GPU-powered servers that may each include a plurality of GPUs, PCIe switches, and/or CPUs, interconnected with high-speed interconnects such as NVLink and PCIe connections. In some examples, a local cache (high-bandwidth scaled out file system) may be available next to the compute cluster and used to cache datasets next to the compute nodes. The system may handle the caching and may provide a local dataset to the compute job. The training apps may produce trained models and experimental metadata that may be stored in a model data store for further consumption.

Model refinement, pruning, and/or fine tuning **130** may include updating the DNNs to further refine and improve the accuracy and efficacy of the DNNs. For example, hyperparameter discovery may be enabled by an experiment service that may track information on the hyper-parameter space to explore hyper-parameter configurations, metrics, and model versions generated by each experiment. A workflow manager may be used to schedule and dispatch experi-

ments to multiple nodes in parallel. Maximum compute efficiency may be enabled by early-termination heuristics which make it possible to terminate experiments that perform poorly relative to other experiments. The workflow manager may be responsible for building binaries out of Source Configuration Management (SCM) repositories, to ensure that no untracked information leaks from users to the generated models. As such, the platform may enable systematic tracking of all information pertaining to experiments, thereby enabling reproducible experiments. Pruning may be executed, as an example and without limitation, similarly to the pruning methods, processes, and systems disclosed in U.S. Provisional Patent Application No. 62/630,445, and U.S. patent application Ser. No. 16/246,414, each hereby incorporated by reference in its entirety.

Model validation 132 may include verifying and/or validating the DNNs, such as by using the validation/verification sub-system 112 of FIG. 1. Once models are validated, a global coverage map may be updated (at 134). For example, once the necessary portions of the software stack(s) 116 are trained for a new region, and/or the software stack(s) 116 for an already covered region are updated, the global coverage map may be updated. As the global coverage map increases, the vehicle(s) 102 using the software stack(s) 116 may be able to navigate through additional regions. Once the models are trained, the models may be reloaded into a larger application and run other test datasets. In such examples, a similar model as used for training may be used for this re-training, testing, or fine tuning process.

In some examples, active learning may be used. For example, existing trained models (e.g. DNNs) may be used to mine for more training data. The system may use existing models to score (or inference) newly collected data and/or raw data and to compute a confidence score for each piece of data. The confidence score may be representative of how informative or useful the data may be for training. For example, data already used that is modeled by an existing model may not provide much or any incremental value, while new data that the model poorly predicts may be reused to improve the model for real-world driving applications. In other words, data that the DNNs are already trained to process accurately may not be as useful as data that the DNNs are not trained to process accurately. Active learning may be used to identify the data that may be used to provide increased performance for the DNNs in additional or alternative situations or environments.

Now referring to FIGS. 3A-3E, FIGS. 3A-3E include workflows used for training DNNs. FIG. 3A includes a workflow 300A. The workflow 300A may include data ingestion 122, passing of the data to dataset store(s) 302 (e.g., a service that handles immutable datasets for further processing), labeling the data using data labeling services 126, and training DNNs using model training 128. The frames selected for labelling may be randomly selected in some examples. The workflow 300A may include labeling of, for example, 300,000 to 600,000 frames (e.g., frames represented by the data). Once the DNNs are trained, the DNNs may be used in simulation and/or re-simulation applications 304. The models may then be pruned, optimized, refined, and then deployed as deployed DNNs in the vehicle(s) 102 (e.g., in the software stack(s) 116). Once the DNNs have been trained to an acceptable level of accuracy (e.g., 90%, 95%, 97%, etc.), the training and refinement process may move to a second workflow, such as workflow 300B.

FIG. 3B includes the workflow 300B. The workflow 300B may include a model store 306 that may contain pre-trained or previously trained models (e.g., DNNs). The pre-trained models may be used to score new data, and the score may be used to prioritize which data to label. For example, a pre-trained DNN may be used to compute a score for each new frame selected where the score may represent a confidence in the prediction of the DNN. When the confidence score is high (e.g., meaning the model is able to accurately handle the frame), the frame may be deprioritized for labeling. When the score is low, the frame may be prioritized. As such, when a frame is confusing for the DNN (i.e., when the confidence score is low), then the frame may be labeled so that the DNN can learn from the frame, thereby further refining the pre-trained DNN.

In some examples, a scoring function may estimate an uncertainty pseudo-probability from the network output. In such examples, dropout may be leveraged by computing an output of the DNN many times, and each time randomly dropping out (e.g., zeroing) neurons of the previous to last layer. The variance of the predictions of the DNN may then be leveraged, and the variance may encode the uncertainty in the prediction, thereby leading to a score.

Key performance indicators (KPIs) and/or metrics may be computed for one or more of the current DNNs (e.g., the best performing DNNs) in the model store 306 in order to determine conditions or combination of conditions which the current DNNs may not perform sufficiently well. For example, one condition dimension may include properties of the whole image or frame, such as, but not limited to, lighting or illumination (e.g., day, night, cloudy, twilight, backlit, etc.), weather (e.g., clear, rain, snow, fog, etc.), setting (e.g., rural, urban, suburban, highway, etc.), topography (e.g., flat, curve, hill, etc.), geographic region (e.g., Europe, North America, China, etc.), sensor (e.g., camera) properties such as position and/or lens type, and/or a combination thereof. The conditions or a combination of the conditions which the current DNNs are not considered to perform sufficiently well on (e.g., have an accuracy below a desired or required level) may be used to direct mining and labeling of data (e.g., additional data) that may increase the accuracy of the DNNs with reference to the conditions or combination of conditions. In some examples, the mining of the data may be facilitated by use tags that may have been added during data indexing and/or curation 124 (FIG. 1).

The workflow 300B may provide for fine-tuning and/or transfer learning. For example, the system may reload models from the model store 306 (as indicated by dashed line 310 in FIG. 3B) and continue to train them. This may be used for camera adaptation and quick experiments, for example.

FIG. 3C includes workflow 300C. The workflow 300C may provide for active learning. For example, the workflows 300A and 300B may not be able to continue indefinitely as this may result in too much data for even the largest systems to retain and process. For example, a data collection of approximately 10,000 hours may result in 30-50 petabytes of data, depending on the number of sensors of the vehicle(s) 102. In addition, after the workflows 300A and 300B, the DNNs should be operating at high accuracy, and performing well on nearly all offline benchmarks. As such, the workflow 300C may provide for edge-level confusion scoring, which may be similar to the workflow 300B, but performed at the edge. In general, this may mean that anything that the DNN fails to understand at the vehicle level while driving may be logged and posted via an API, and flagged for further inspection (e.g., labeling).

In some examples, pre-determined conditions or combinations of conditions (such as those described herein) that

the DNNs fail to perform accurately enough may also be used to direct and focus data gathering at the edge. For example, the vehicle(s) 102 may not collect all data, but may only collect data where certain conditions or combinations of conditions are met (e.g., at night, in the rain, in certain tunnel types, etc.).

FIG. 3D includes workflow 300D. The workflow 300D may provide for training and refinement of the DNNs. In some examples, the workflow 300D may follow the workflow 300C. Ultimately, the goal may be to use prediction scores and aggregate them on a map (e.g., a GPS or GNSS map), to illustrate locations or regions where the DNNs perform well, and where the DNNs perform less well (e.g., below desired accuracy levels). A heat map 312 may be generated to indicate the areas of lower performance, and the vehicle(s) 102 may be dispatched (e.g., at dispatch 308) to these locations. In some examples, the KPIs and/or metrics may be leveraged to direct planning of routes for the vehicle(s) 102 in order to capture data representative of conditions or combinations of conditions that the DNN is not as accurate at generating predictions for.

Simulation System

The simulation system 400—e.g., represented by simulation systems 400A, 400B, 400C, and 400D, described in more detail herein—may generate a global simulation that simulates a virtual world or environment (e.g., a simulated environment) that may include artificial intelligence (AI) vehicles or other objects (e.g., pedestrians, animals, etc.), hardware-in-the-loop (HIL) vehicles or other objects, software-in-the-loop (SIL) vehicles or other objects, and/or person-in-the-loop (PIL) vehicles or other objects. The global simulation may be maintained within an engine (e.g., a game engine), or other software-development environment, that may include a rendering engine (e.g., for 2D and/or 3D graphics), a physics engine (e.g., for collision detection, collision response, etc.), sound, scripting, animation, AI, networking, streaming, memory management, threading, localization support, scene graphs, cinematics, and/or other features. In some examples, as described herein, one or more vehicles or objects within the simulation system 400 (e.g., HIL objects, SIL objects, PIL objects, AI objects, etc.) may be maintained within their own instance of the engine. In such examples, each virtual sensor of each virtual object may include their own instance of the engine (e.g., an instance for a virtual camera, a second instance for a virtual LIDAR sensor, a third instance for another virtual LIDAR sensor, etc.). As such, an instance of the engine may be used for processing sensor data for each sensor with respect to the sensor's perception of the global simulation. As such, for a virtual camera, the instance may be used for processing image data with respect to the camera's field of view in the simulated environment. As another example, for an IMU sensor, the instance may be used for processing IMU data (e.g., representative of orientation) for the object in the simulated environment.

AI (e.g., bots) vehicles or other objects may include pedestrians, animals, third-party vehicles, vehicles, and/or other object types. The AI objects in the simulated environment may be controlled using artificial intelligence (e.g., machine learning such as neural networks, rules-based control, a combination thereof, etc.) in a way that simulates, or emulates, how corresponding real-world objects would behave. In some examples, the rules, or actions, for AI objects may be learned from one or more HIL objects, SIL objects, and/or PIL objects. In an example where an AI object (e.g., bot) in the simulated environment corresponds to a pedestrian, the bot may be trained to act like a pedestrian in any of a number of different situations or environments (e.g., running, walking, jogging, not paying attention, on the phone, raining, snowing, in a city, in a suburban area, in a rural community, etc.). As such, when the simulated environment is used for testing vehicle performance (e.g., for HIL or SIL embodiments), the bot (e.g., the pedestrian) may behave as a real-world pedestrian would (e.g., by jaywalking in rainy or dark conditions, failing to heed stop signs or traffic lights, etc.), in order to more accurately simulate a real-world environment. This method may be used for any AI bot in the simulated environment, such as vehicles, bicyclists, or motorcycles, whose AI bots may also be trained to behave as real-world objects would (e.g., weaving in and out of traffic, swerving, changing lanes with no signal or suddenly, braking unexpectedly, etc.).

The AI objects that may be distant from the vehicle of interest (e.g., the ego-vehicle in the simulated environment) may be represented in a simplified form—such as a radial distance function, or list of points at known positions in a plane, with associated instantaneous motion vectors. As such, the AI objects may be modeled similarly to how AI agents may be modeled in videogame engines.

HIL vehicles or objects may use hardware that is used in the physical vehicles or objects to at least assist in some of the control of the HIL vehicles or objects in the simulated environment. For example, a vehicle controlled in a HIL environment may use one or more SoCs 1104 (FIG. 11C), CPU(S) 1118, GPU(s) 1120, etc. in a data flow loop for controlling the vehicle in the simulated environment. In some examples, the hardware from the vehicles may be an NVIDIA DRIVE AGX Pegasus™ compute platform and/or an NVIDIA DRIVE PX Xavier™ compute platform. For example, the vehicle hardware (e.g., vehicle hardware 104) may include some or all of the components and/or functionality described in U.S. Non-Provisional application Ser. No. 16/186,473, filed on Nov. 9, 2018, which is hereby incorporated by reference in its entirety. In such examples, at least some of the control decisions may be generated using the hardware that is configured for installation within a real-world autonomous vehicle (e.g., the vehicle 102) to execute at least a portion of a software stack(s) 116 (e.g., an autonomous driving software stack).

SIL vehicles or objects may use software to simulate or emulate the hardware from the HIL vehicles or objects. For example, instead of using the actual hardware that may be configured for use in physical vehicles (e.g., the vehicle 102), software, hardware, or a combination thereof may be used to simulate or emulate the actual hardware (e.g., simulate the SoC(s) 1104).

PIL vehicles or objects may use one or more hardware components that allow a remote operator (e.g., a human, a robot, etc.) to control the PIL vehicle or object within the simulated environment. For example, a person or robot may control the PIL vehicle using a remote control system (e.g., including one or more pedals, a steering wheel, a VR system, etc.), such as the remote control system described in U.S. Non-Provisional application Ser. No. 16/366,506, filed on Mar. 27, 2019, and hereby incorporated by reference in its entirety. In some examples, the remote operator may control autonomous driving level 0, 1, or 2 (e.g., according to the Society of Automotive Engineers document J3016) virtual vehicles using a VR headset and a CPU(s) (e.g., an X86 processor), a GPU(s), or a combination thereof. In other examples, the remote operator may control advanced AI-assisted level 2, 3, or 4 vehicles modeled using one or more advanced SoC platforms. In some examples, the PIL vehicles or objects may be recorded and/or tracked, and the

recordings and/or tracking data may be used to train or otherwise at least partially contribute to the control of AI objects, such as those described herein.

Now referring to FIG. 4A, FIG. 4A is an example illustration of a simulation system 400A, in accordance with some embodiments of the present disclosure. The simulation system 400A may generate a simulated environment 410 that may include AI objects 412 (e.g., AI objects 412A and 412B), HIL objects 414, SIL objects 416, PIL objects 418, and/or other object types. The simulated environment 410 may include features of a driving environment, such as roads, bridges, tunnels, street signs, stop lights, crosswalks, buildings, trees and foliage, the sun, the moon, reflections, shadows, etc., in an effort to simulate a real-world environment accurately within the simulated environment 410. In some examples, the features of the driving environment within the simulated environment 410 may be more true-to-life by including chips, paint, graffiti, wear and tear, damage, etc. Although described with respect to a driving environment, this is not intended to be limiting, and the simulated environment may include an indoor environment (e.g., for a robot, a drone, etc.), an aerial environment (e.g., for a UAV, a drone, an airplane, etc.), an aquatic environment (e.g., for a boat, a ship, a submarine, etc.), and/or another environment type.

The simulated environment 410 may be generated using virtual data, real-world data, or a combination thereof. For example, the simulated environment may include real-world data augmented or changed using virtual data to generate combined data that may be used to simulate certain scenarios or situations with different and/or added elements (e.g., additional AI objects, environmental features, weather conditions, etc.). For example, pre-recorded video may be augmented or changed to include additional pedestrians, obstacles, and/or the like, such that the virtual objects (e.g., executing the software stack(s) 116 as HIL objects and/or SIL objects) may be tested against variations in the real-world data.

The simulated environment may be generated using rasterization, ray-tracing, using DNNs such as generative adversarial networks (GANs), another rendering technique, and/or a combination thereof. For example, in order to create more true-to-life, realistic lighting conditions (e.g., shadows, reflections, glare, global illumination, ambient occlusion, etc.), the simulation system 400A may use real-time ray-tracing. In one or more embodiments, one or more hardware accelerators may be used by the simulation system 400A to perform real-time ray-tracing. The ray-tracing may be used to simulate LIDAR sensor for accurate generation of LIDAR data. For example, ray casting may be used in an effort to simulate LIDAR reflectivity. In any example, ray-tracing techniques used by the simulation system 400A may include one or more techniques described in U.S. Provisional Patent Application No. 62/644,385, filed Mar. 17, 2018, U.S. Provisional Patent Application No. 62/644,386, filed Mar. 17, 2018, U.S. Provisional Patent Application No. 62/644, 601, filed Mar. 19, 2018, and U.S. Provisional Application No. 62/644,806, filed Mar. 19, 2018, U.S. Non-Provisional patent application Ser. No. 16/354,983, filed on Mar. 15, 2019, and/or U.S. Non-Provisional patent application Ser. No. 16/355,214, filed on Mar. 15, 2019, each of which is hereby incorporated by reference in its entirety.

In some examples, the simulated environment may be rendered, at least in part, using one or more DNNs, such as generative adversarial neural networks (GANs). For example, real-world data may be collected, such as real-world data captured by autonomous vehicles (e.g.,

camera(s), LIDAR sensor(s), RADAR sensor(s), etc.), robots, and/or other objects, as well as real-world data that may be captured by any sensors (e.g., images or video pulled from data stores, online resources such as search engines, etc.). The real-world data may then be segmented, classified, and/or categorized, such as by labeling differing portions of the real-world data based on class (e.g., for an image of a landscape, portions of the image—such as pixels or groups of pixels—may be labeled as car, sky, tree, road, building, water, waterfall, vehicle, bus, truck, sedan, etc.). A GAN (or other DNN) may then be trained using the segmented, classified, and/or categorized data to generate new versions of the different types of objects, landscapes, and/or other features as graphics within the simulated environment.

The simulator component(s) 402 of the simulation system 400 may communicate with vehicle simulator component(s) 406 over a wired and/or wireless connection. In some examples, the connection may be a wired connection using one or more sensor switches 408, where the sensor switches may provide low-voltage differential signaling (LVDS) output. For example, the sensor data (e.g., image data) may be transmitted over an HDMI to LVDS connection between the simulator component(s) 402 and the vehicle simulator component(s) 406. The simulator component(s) 402 may include any number of compute nodes (e.g., computers, servers, etc.) interconnected in order to ensure synchronization of the world state. In some examples, as described herein, the communication between each of the compute nodes (e.g., the vehicle simulator component(s) compute nodes and the simulator component(s) compute nodes) may be managed by a distributed shared memory (DSM) system (e.g., DSM 424 of FIG. 4C) using a distributed shared memory protocol (e.g., a coherence protocol). The DSM may include a combination of hardware (cache coherence circuits, network interfaces, etc.) and software. This shared memory architecture may separate memory into shared parts distributed among nodes and main memory, or distributing all memory between all nodes. In some examples, InfiniBand (IB) interfaces and associated communications standards may be used. For example, the communication between and among different nodes of the simulation system 400 (and/or 600) may use IB.

The simulator component(s) 402 may include one or more GPUs 404. The virtual vehicle being simulated may include any number of sensors (e.g., virtual or simulated sensors) that may correspond to one or more of the sensors described herein at least with respect to FIGS. 11A-11C. In some examples, each sensor of the vehicle may correspond to, or be hosted by, one of the GPUs 404. For example, processing for a LIDAR sensor may be executed on a first GPU 404, processing for a wide-view camera may be executed on a second GPU 404, processing for a RADAR sensor may be executed on a third GPU, and so on. As such, the processing of each sensor with respect to the simulated environment may be capable of executing in parallel with each other sensor using a plurality of GPUs 404 to enable real-time simulation. In other examples, two or more sensors may correspond to, or be hosted by, one of the GPUs 404. In such examples, the two or more sensors may be processed by separate threads on the GPU 404 and may be processed in parallel. In other examples, the processing for a single sensor may be distributed across more than one GPU. In addition to, or alternatively from, the GPU(s) 404, one or more TPUs, CPUs, and/or other processor types may be used for processing the sensor data.

Vehicle simulator component(s) 406 may include a compute node of the simulation system 400A that corresponds to

a single vehicle represented in the simulated environment **410**. Each other vehicle (e.g., **414**, **418**, **416**, etc.) may include a respective node of the simulation system. As a result, the simulation system **400A** may be scalable to any number of vehicles or objects as each vehicle or object may be hosted by, or managed by, its own node in the system **400A**. In the illustration of FIG. **4A**, the vehicle simulator component(s) **406** may correspond to a HIL vehicle (e.g., because the vehicle hardware **104** is used). However, this is not intended to be limiting and, as illustrated in FIGS. **4B** and **4C**, the simulation system **400** may include SIL vehicles, HIL vehicles, PIL vehicles, and/or AI vehicles. The simulator component(s) **402** (e.g., simulator host device) may include one or more compute nodes of the simulation system **400A**, and may host the simulation of the environment with respect to each actor (e.g., with respect to each HIL, SIL, PIL, and AI actors), as well as hosting the rendering and management of the environment or world state (e.g., the road, signs, trees, foliage, sky, sun, lighting, etc.). In some examples, the simulator component(s) **402** may include a server(s) and associated components (e.g., CPU(s), GPU(s), computers, etc.) that may host a simulator (e.g., NVIDIA's DRIVE™ Constellation AV Simulator).

The vehicle hardware **104**, as described herein, may correspond to the vehicle hardware **104** of FIG. **1** that may be used in the physical vehicle **102**. However, in the simulation system **400A**, the vehicle hardware **104** may be incorporated into the vehicle simulator component(s) **406**. As such, because the vehicle hardware **104** may be configured for installation within the vehicle **102**, the simulation system **400A** may be specifically configured to use the vehicle hardware **104** within a node (e.g., of a server platform) of the simulation system **400A**. For example, similar interfaces used in the physical vehicle **102** may need to be used by the vehicle simulator component(s) **406** to communicate with the vehicle hardware **104**. In some examples, the interfaces may include: (1) CAN interfaces, including a PCAN adapter, (2) Ethernet interfaces, including RAW UDP sockets with IP address, origin, VLA, and/or source IP all preserved, (3) Serial interfaces, with a USB to serial adapter, (4) camera interfaces, (5) InfiniBand (IB) interfaces, and/or other interface types.

In any examples, once the sensor data representative of a field(s) of view of the sensor(s) of the vehicle in the simulated environment has been generated and/or processed (e.g., using one or more codecs, as described herein), the sensor data (and/or encoded sensor data) may be used by the software stack(s) **116** (e.g., the autonomous driving software stack) executed on the vehicle hardware **104** to perform one or more operations (e.g., generate one or more controls, route planning, detecting objects, identifying drivable freespace, monitoring the environment for obstacle avoidance, etc.). As a result, the identical, or substantially identical, hardware components used by the vehicle **102** (e.g., a physical vehicle) to execute the autonomous driving software stack in real-world environments may be used to execute the autonomous driving software stack in the simulated environment **410**. The use of the vehicle hardware **104** in the simulation system **400A** thus provides for a more accurate simulation of how the vehicle **102** will perform in real-world situations, scenarios, and environments without having to actually find and test the vehicle **102** in the real-world. This may reduce the amount of driving time required for testing the hardware/software combination used in the physical vehicle **102** and may reduce safety risks by not requiring actual real-world testing (especially for dangerous situations, such as other vehicles driving erratically or at unsafe speeds, children playing in the street, ice on a bridge, etc.).

In addition to the vehicle hardware **104**, the vehicle simulator component(s) **406** may manage the simulation of the vehicle (or other object) using additional hardware, such as a computer—e.g., an X86 box. In some examples, additional processing for virtual sensors of the virtual object may be executed using the vehicle simulation component(s) **406**. In such examples, at least some of the processing may be performed by the simulator component(s) **402**, and other of the processing may be executed by the vehicle simulator component(s) **406** (or **420**, or **422**, as described herein). In other examples, the processing of the virtual sensors may be executed entirely on the vehicle simulator component(s) **406**.

Now referring to FIG. **4B**, FIG. **4B** is another example illustration of a simulation system **400B**, in accordance with some embodiments of the present disclosure. The simulation system **400B** may include the simulator component(s) **402** (as one or more compute nodes), the vehicle simulator component(s) **406** (as one or more compute nodes) for a HIL object(s), the vehicle simulator component(s) **420** (as one or more compute nodes) for a SIL object(s), the vehicle simulator component(s) **406** (as one or more compute nodes) for a PIL object(s), and/or additional component(s) (or compute nodes) for AI objects and/or other object types. Each of the PIL, HIL, SIL, AI, and/or other object type compute nodes may communicate with the simulator component(s) **402** to capture from the global simulation at least data that corresponds to the respective object within the simulate environment **410**.

For example, the vehicle simulator component(s) **422** may receive (e.g., retrieve, obtain, etc.), from the global simulation (e.g., represented by the simulated environment **410**) hosted by the simulator component(s) **402**, data that corresponds to, is associated with, and/or is required by the vehicle simulator component(s) **422** to perform one or more operations by the vehicle simulator component(s) **422** for the PIL object. In such an example, data (e.g., virtual sensor data corresponding to a field(s) of view of virtual camera(s) of the virtual vehicle, virtual LIDAR data, virtual RADAR data, virtual location data, virtual IMU data, etc.) corresponding to each sensor of the PIL object may be received from the simulator component(s) **402**. This data may be used to generate an instance of the simulated environment corresponding to the field of view of a remote operator of the virtual vehicle controlled by the remote operator, and the portion of the simulated environment may be projected on a display (e.g., a display of a VR headset, a computer or television display, etc.) for assisting the remote operator in controlling the virtual vehicle through the simulated environment **410**. The controls generated or input by the remote operator using the vehicle simulator component(s) **422** may be transmitted to the simulator component(s) **402** for updating a state of the virtual vehicle within the simulated environment **410**.

As another example, the vehicle simulator component(s) **420** may receive (e.g., retrieve, obtain, etc.), from the global simulation hosted by the simulator component(s) **402**, data that corresponds to, is associated with, and/or is required by the vehicle simulator component(s) **420** to perform one or more operations by the vehicle simulator component(s) **420** for the SIL object. In such an example, data (e.g., virtual sensor data corresponding to a field(s) of view of virtual camera(s) of the virtual vehicle, virtual LIDAR data, virtual RADAR data, virtual location data, virtual IMU data, etc.)

corresponding to each sensor of the SIL object may be received from the simulator component(s) **402**. This data may be used to generate an instance of the simulated environment for each sensor (e.g., a first instance from a field of view of a first virtual camera of the virtual vehicle, a second instance from a field of view of a second virtual camera, a third instance from a field of view of a virtual LIDAR sensor, etc.). The instances of the simulated environment may thus be used to generate sensor data for each sensor by the vehicle simulator component(s) **420**. In some examples, the sensor data may be encoded using one or more codecs (e.g., each sensor may use its own codec, or each sensor type may use its own codec) in order to generate encoded sensor data that may be understood or familiar to an autonomous driving software stack simulated or emulated by the vehicle simulator component(s) **420**. For example, a first vehicle manufacturer may use a first type of LIDAR data, a second vehicle manufacturer may use a second type of LIDAR data, etc., and thus the codecs may customize the sensor data to the types of sensor data used by the manufacturers. As a result, the simulation system **400** may be universal, customizable, and/or useable by any number of different sensor types depending on the types of sensors and the corresponding data types used by different manufacturers. In any example, the sensor data and/or encoded sensor data may be used by an autonomous driving software stack to perform one or more operations (e.g., object detection, path planning, control determinations, actuation types, etc.). For example, the sensor data and/or encoded data may be used as inputs to one or more DNNs of the autonomous driving software stack, and the outputs of the one or more DNNs may be used for updating a state of the virtual vehicle within the simulated environment **410**. As such, the reliability and efficacy of the autonomous driving software stack, including one or more DNNs, may be tested, fine-tuned, verified, and/or validated within the simulated environment.

In yet another example, the vehicle simulator component(s) **406** may receive (e.g., retrieve, obtain, etc.), from the global simulation hosted by the simulator component(s) **402**, data that corresponds to, is associated with, and/or is required by the vehicle simulator component(s) **406** to perform one or more operations by the vehicle simulator component(s) **406** for the HIL object. In such an example, data (e.g., virtual sensor data corresponding to a field(s) of view of virtual camera(s) of the virtual vehicle, virtual LIDAR data, virtual RADAR data, virtual location data, virtual IMU data, etc.) corresponding to each sensor of the HIL object may be received from the simulator component(s) **402**. This data may be used to generate an instance of the simulated environment for each sensor (e.g., a first instance from a field of view of a first virtual camera of the virtual vehicle, a second instance from a field of view of a second virtual camera, a third instance from a field of view of a virtual LIDAR sensor, etc.). The instances of the simulated environment may thus be used to generate sensor data for each sensor by the vehicle simulator component(s) **420**. In some examples, the sensor data may be encoded using one or more codecs (e.g., each sensor may use its own codec, or each sensor type may use its own codec) in order to generate encoded sensor data that may be understood or familiar to an autonomous driving software stack executing on the vehicle hardware **104** of the vehicle simulator component(s) **420**. Similar to the SIL object described herein, the sensor data and/or encoded sensor data may be used by an autonomous driving software stack to perform one or more operations (e.g., object detection, path planning, control determinations, actuation types, etc.).

Now referring to FIG. 4C, FIG. 4C is another example illustration of a simulation system **400C**, in accordance with some embodiments of the present disclosure. The simulation system **400C** may include distributed shared memory (DSM) system **242**, the simulator component(s) **402** (as one or more compute nodes), the vehicle simulator component(s) **406** (as one or more compute nodes) for a HIL object(s), the vehicle simulator component(s) **420** (as one or more compute nodes) for a SIL object(s), the vehicle simulator component(s) **406** (as one or more compute nodes) for a PIL object(s), and/or additional component(s) (or compute nodes) for AI objects and/or other object types (not shown). The simulation system **400C** may include any number of HIL objects (e.g., each including its own vehicle simulator component(s) **406**), any number of SIL objects (e.g., each including its own vehicle simulator component(s) **420**), any number of PIL objects (e.g., each including its own vehicle simulator component(s) **422**), and/or any number of AI objects (not shown, but may be hosted by the simulation component(s) **402** and/or separate compute nodes, depending on the embodiment).

The vehicle simulator component(s) **406** may include one or more SoC(s) **1104** (or other components) that may be configured for installation and use within a physical vehicle. As such, as described herein, the simulation system **400C** may be configured to use the SoC(s) **1104** and/or other vehicle hardware **104** by using specific interfaces for communicating with the SoC(s) **1104** and/or other vehicle hardware. The vehicle simulator component(s) **420** may include one or more software instances **430** that may be hosted on one or more GPUs and/or CPUs to simulate or emulate the SoC(s) **1104**. The vehicle simulator component(s) **422** may include one or more SoC(s) **426**, one or more CPU(s) **428** (e.g., X86 boxes), and/or a combination thereof, in addition to the component(s) that may be used by the remote operator (e.g., keyboard, mouse, joystick, monitors, VR systems, steering wheel, pedals, in-vehicle components, such as light switches, blinkers, HMI display(s), etc., and/or other component(s)).

The simulation component(s) **402** may include any number of CPU(s) **432** (e.g., X86 boxes), GPU(s), and/or a combination thereof. The CPU(s) **432** may host the simulation software for maintaining the global simulation, and the GPU(s) **434** may be used for rendering, physics, and/or other functionality for generating the simulated environment **410**.

As described herein, the simulation system **400C** may include the DSM **424**. The DSM **424** may use one or more distributed shared memory protocols to maintain the state of the global simulation using the state of each of the objects (e.g., HIL objects, SIL objects, PIL objects, AI objects, etc.). As such, each of the compute nodes corresponding to the vehicle simulator component(s) **406**, **420**, and/or **422** may be in communication with the simulation component(s) **402** via the DSM **424**. By using the DSM **424** and the associated protocols, real-time simulation may be possible. For example, as opposed to how network protocols (e.g., TCP, UDP, etc.) are used in massive multiplayer online (MMO) games, the simulation system **400** may use a distributed shared memory protocol to maintain the state of the global simulation and each instance of the simulation (e.g., by each vehicle, object, and/or sensor) in real-time.

Now referring to FIG. 4D, FIG. 4D is an example illustration of a hardware-in-the-loop configuration, in accordance with some embodiments of the present disclosure. The vehicle simulator component(s) **406** may include the vehicle hardware **104**, as described herein, and may

include one or more computer(s) **436**, one or more GPU(s) (not shown), and/or one or more CPU(s) (not shown). The computer(s) **436**, GPU(s), and/or CPU(s) may manage or host the simulation software **438**, or instance thereof, executing on the vehicle simulator component(s) **406**. The vehicle hardware **104** may execute the software stack(s) **116** (e.g., an autonomous driving software stack, an IX software stack, etc.).

As described herein, by using the vehicle hardware **104**, the other vehicle simulator component(s) **406** within the simulation environment **400** may need to be configured for communication with the vehicle hardware **104**. For example, because the vehicle hardware **104** may be configured for installation within a physical vehicle (e.g., the vehicle **102**), the vehicle hardware **104** may be configured to communicate over one or more connection types and/or communication protocols that are not standard in computing environments (e.g., in server-based platforms, in general-purpose computers, etc.). For example, a CAN interface, LVDS interface, USB interface, Ethernet interface, InfiniBand (IB) interface, and/or other interfaces may be used by the vehicle hardware **104** to communicate signals with other components of the physical vehicle. As such, in the simulation system **400**, the vehicle simulator component(s) **406** (and/or other component(s) of the simulation system **400** in addition to, or alternative from, the vehicle simulator component(s) **406**) may need to be configured for use with the vehicle hardware **104**. In order to accomplish this, one or more CAN interfaces, LVDS interfaces, USB interfaces, Ethernet interfaces, and/or other interface may be used to provide for communication (e.g., over one or more communication protocols, such as LVDS) between vehicle hardware **104** and the other component(s) of the simulation system **400**.

In some examples, the virtual vehicle that may correspond to the vehicle simulator component(s) **406** within the simulation system **400** may be modeled as a game object within an instance of a game engine. In addition, each of the virtual sensors of the virtual vehicle may be interfaced using sockets within the virtual vehicle's software stack(s) **116** executed on the vehicle hardware **104**. In some examples, each of the virtual sensors of the virtual vehicle may include an instance of the game engine, in addition to the instance of the game engine associated with the simulation software **438** for the virtual vehicle. In examples where the vehicle simulator component(s) **406** include a plurality of GPUs, each of the sensors may be executed on a single GPU. In other examples, multiple sensors may be executed on a single GPU, or at least as many sensors as feasible to ensure real-time generation of the virtual sensor data.

Using HIL objects in the simulator system **400** may provide for a scalable solution that may simulate or emulate various driving conditions for autonomous software and hardware systems (e.g., NVIDIA's DRIVE AGX Pegasus™ compute platform and/or DRIVE PX Xavier™ compute platform). Some benefits of HIL objects may include the ability to test DNNs faster than real-time, the ability to scale verification with computing resources (e.g., rather than vehicles or test tracks), the ability to perform deterministic regression testing (e.g., the real-world environment is never the same twice, but a simulated environment can be), optimal ground truth labeling (e.g., no hand-labeling required), the ability to test scenarios difficult to produce in the real-world, rapid generation of test permutations, and the ability to test a larger space of permutations in simulation as compared to real-world.

Now referring to FIG. 4E, FIG. 4E is an example illustration of a hardware-in-the-loop configuration, in accor-

dance with some embodiments of the present disclosure. The HIL configuration of FIG. 4E may include vehicle simulator component(s) **406**, including the SoC(s) **1104**, a chassis fan(s) **456** and/or water-cooling system. The HIL configuration may include a two-box solution (e.g., the simulator component(s) **402** in a first box and the vehicle simulator component(s) **406** in a second box). Using this approach may reduce the amount of space the system occupies as well as reduce the number of external cables in data centers (e.g., by including multiple components together with the SoC(s) **1104** in the vehicle simulator component(s) **406**—e.g., the first box). The vehicle simulator component(s) **406** may include one or more GPUs **452** (e.g., NVIDIA QUADRO GPU(s)) that may provide, in an example, non-limiting embodiment, 8 DP/HDMI video streams that may be synchronized using sync component(s) **454** (e.g., through a QUADRO Sync II Card). These GPU(s) **452** (and/or other GPU types) may provide the sensor input to the SoC(s) **1104** (e.g., to the vehicle hardware **104**). In some examples, the vehicle simulator component(s) **406** may include a network interface (e.g., one or more network interface cards (NICs) **450**) that may simulate or emulate RADAR sensors, LIDAR sensors, and/or IMU sensors (e.g., by providing 8 Gigabit ports with precision time protocol (PTP) support). In addition, the vehicle simulator component(s) **406** may include an input/output (I/O) analog integrated circuit. Registered Jack (RJ) interfaces (e.g., RJ45), high speed data (HSD) interfaces, USB interfaces, pulse per second (PPS) clocks, Ethernet (e.g., 10 Gb Ethernet (GbE)) interfaces, CAN interfaces, HDMI interfaces, and/or other interface types may be used to effectively transmit and communication data between and among the various component(s) of the system.

Now referring to FIG. 4F, FIG. 4F is an example illustration of a software-in-the-loop configuration, in accordance with some embodiments of the present disclosure. The vehicle simulator component(s) **420** may include computer(s) **440**, GPU(s) (not shown), CPU(s) (not shown), and/or other components. The computer(s) **440**, GPU(s), and/or CPU(s) may manage or host the simulation software **438**, or instance thereof, executing on the vehicle simulator component(s) **420**, and may host the software stack(s) **116**. For example, the vehicle simulator component(s) **420** may simulate or emulate, using software, the vehicle hardware **104** in an effort to execute the software stack(s) **116** as accurately as possible.

In order to increase accuracy in SIL embodiments, the vehicle simulator component(s) **420** may be configured to communicate over one or more virtual connection types and/or communication protocols that are not standard in computing environments. For example, a virtual CAN interface, virtual LVDS interface, virtual USB interface, virtual Ethernet interface, and/or other virtual interfaces may be used by the computer(s) **440**, CPU(s), and/or GPU(s) of the vehicle simulator component(s) **420** to provide for communication (e.g., over one or more communication protocols, such as LVDS) between the software stack(s) **116** and the simulation software **438** within the simulation system **400**. For example, the virtual interfaces may include middleware that may be used to provide a continuous feedback loop with the software stack(s) **116**. As such, the virtual interfaces may simulate or emulate the communications between the vehicle hardware **104** and the physical vehicle using one or more software protocols, hardware (e.g., CPU(s), GPU(s), computer(s) **440**, etc.), or a combination thereof.

The computer(s) **440** in some examples, may include X86 CPU hardware, and one or more X86 CPUs may execute both the simulation software **438** and the software stack(s)

116. In other examples, the computer(s) **440** may include GPU hardware (e.g., an NVIDIA DGX system and/or cloud-based NVIDIA Tesla servers).

In some examples, the virtual vehicle that may correspond to the vehicle simulator component(s) **420** within the simulation system **400** may be modeled as a game object within an instance of a game engine. In addition, each of the virtual sensors of the virtual vehicle may be interfaced using sockets within the virtual vehicle's software stack(s) **116** executed on the vehicle simulator component(s) **420**. In some examples, each of the virtual sensors of the virtual vehicle may include an instance of the game engine, in addition to the instance of the game engine associated with the simulation software **438** for the virtual vehicle. In examples where the vehicle simulator component(s) **406** include a plurality of GPUs, each of the sensors may be executed on a single GPU. In other examples, multiple sensors may be executed on a single GPU, or at least as many sensors as feasible to ensure real-time generation of the virtual sensor data.

Now referring to FIG. **5**, each block of method **500**, described herein, comprises a computing process that may be performed using any combination of hardware, firmware, and/or software. For instance, various functions may be carried out by a processor executing instructions stored in memory. The method may also be embodied as computer-usable instructions stored on computer storage media. The method may be provided by a standalone application, a service or hosted service (standalone or in combination with another hosted service), or a plug-in to another product, to name a few. In addition, method **500** is described, by way of example, with respect to the simulation system **400** of FIGS. 4A-4C. However, the method may additionally or alternatively be executed by any one system, or any combination of systems, including, but not limited to, those described herein.

FIG. **5** is a flow diagram showing a method **500** for generating a simulated environment using a hardware-in-the-loop object, in accordance with some embodiments of the present disclosure. The method **500**, at block B**502**, includes transmitting, from a first hardware component to a second hardware component, simulation data. For example, simulation component(s) **402** may transmit simulation data to one or more of the vehicle simulator component(s) **406**, the vehicle simulator component(s) **420**, and/or the vehicle simulator component(s) **422**. In some examples, the simulation data may be representative of at least a portion of the simulated environment **410** hosted by the simulation component(s) **402**, and may correspond to the simulated environment **410** with respect to at least one virtual sensor of a virtual object (e.g., a HIL object, a SIL object, a PIL object, and/or an AI object). In an example where the virtual sensor is a virtual camera, the simulation data may correspond to at least the data from the simulation necessary to generate a field of view of the virtual camera within the simulated environment **410**.

The method **500**, at block B**504**, includes receiving a signal by the first hardware component and from the second hardware component. For example, the simulator component(s) **402** may receive a signal from one of the vehicle simulator component(s) **406**, the vehicle simulator component(s) **420**, and/or the vehicle simulator component(s) **422**. The signal may be representative of an operation (e.g., control, path planning, object detection, etc.) corresponding to a virtual object (e.g., a HIL object, a SIL object, a PIL object, and/or an AI object) as determined by a software stack(s) **116**. In some examples, such as where the virtual

object is a HIL object, the signal (or data represented thereby) may be transmitted from the vehicle hardware **104** to one or more other vehicle simulator component(s) **406**, and then the vehicle simulator component(s) **406** may transmit the signal to the simulator component(s) **402**. In such examples, the signals between the vehicle simulator component(s) **406** (e.g., between the vehicle hardware **104** and one or more GPU(s), CPU(s), and/or computer(s) **436**) may be transmitted via a CAN interface, a USB interface, an LVDS interface, an Ethernet interface, and/or another interface. In another example, such as where the virtual object is a SIL object, the signal (or data represented thereby) may be transmitted from the vehicle simulator component(s) **420** to the simulator component(s) **402**, where the data included in the signal may be generated by the software stack(s) **116** executing on simulated or emulated vehicle hardware **104**. In such examples, the vehicle simulator component(s) **420** may use a virtual CAN, a virtual LVDS interface, a virtual USB interface, a virtual Ethernet interface, and/or other virtual interfaces.

The method **500**, at block B**506**, includes updating, by the first hardware component, one or more attributes of a virtual object within a simulated environment. For example, based at least in part on the signal received from the vehicle simulator component(s) **406**, the vehicle simulator component(s) **420**, and/or the vehicle simulator component(s) **422**, the simulator component(s) **402** may update the global simulation (and the simulated environment may be updated accordingly). In some examples, the data represented by the signal may be used to update a location, orientation, speed, and/or other attributes of the virtual object hosted by the vehicle simulator component(s) **406**, the vehicle simulator component(s) **420**, and/or the vehicle simulator component(s) **422**.

Now referring to FIG. **6**A, FIG. **6**A is an example illustration of a simulation system **600** at runtime, in accordance with some embodiments of the present disclosure. Some or all of the components of the simulation system **600** may be used in the simulation system **400**, and some or all of the components of the simulation system **400** may be used in the simulation system **600**. As such, components, features, and/or functionality described with respect to the simulation system **400** may be associated with the simulation system **600**, and vice versa. In addition, each of the simulation systems **600**A and **600**B (FIG. **6**B) may include similar and/or shared components, features, and/or functionality.

The simulation system **600**A (e.g., representing one example of simulation system **600**) may include the simulator component(s) **402**, codec(s) **614**, content data store(s) **602**, scenario data store(s) **604**, vehicle simulator component(s) **420** (e.g., for a SIL object), and vehicle simulator component(s) **406** (e.g., for a HIL object). The content data store(s) **602** may include detailed content information for modeling cars, trucks, people, bicyclists, signs, buildings, trees, curbs, and/or other features of the simulated environment. The scenario data store(s) **604** may include scenario information that may include dangerous scenario information (e.g., that is unsafe to test in the real-world environment), such as a child in an intersection.

The simulator component(s) **402** may include an AI engine **608** that simulates traffic, pedestrians, weather, and/or other AI features of the simulated environment. The simulator component(s) **402** may include a virtual world manager **610** that manages the world state for the global simulation. The simulator component(s) **402** may further include a virtual sensor manger **612** that may mange the virtual sensors. The AI engine **608** may model traffic similar

to how traffic is modeled in an automotive video game, and may be done using a game engine, as described herein. In other examples, custom AI may be used to provide the determinism and computational level of detail necessary for large-scale reproducible automotive simulation. In some examples, traffic may be modeled using SIL objects, HIL objects, PIL objects, AI objects, and/or combination thereof. The system **600** may create a subclass of an AI controller that examines map data, computes a route, and drives the route while avoiding other cars. The AI controller may compute desired steering, acceleration, and/or braking, and may apply those values to the virtual objects. The vehicle properties used may include mass, max RPM, torque curves, and/or other properties. A physics engine may be used to determine states of AI objects. As described herein, for vehicles or other objects that may be far away and may not have an impact on a current sensor(s), the system may choose not to apply physics for those objects and only determine locations and/or instantaneous motion vectors. Ray-casting may be used for each wheel to ensure that the wheels of the vehicles are in contact. In some examples, traffic AI may operate according to a script (e.g., rules-based traffic). Traffic AI maneuvers for virtual objects may include lateral lane changes (e.g., direction, distance, duration, shape, etc.), longitudinal movement (e.g., matching speed, relative target, delta to target, absolute value), route following, and/or path following. The triggers for the traffic AI maneuvers may be time-based (e.g., three seconds), velocity-based (e.g., at sixty mph), proximity-based to map (e.g., within twenty feet of intersection), proximity-based to actor (e.g., within twenty feet of another object), lane clear, and/or others.

The AI engine **608** may model pedestrian AI similar to traffic AI, described herein, but for pedestrians. The pedestrians may be modeled similar to real pedestrians, and the system **600** may infer pedestrian conduct based on learned behaviors.

The simulator component(s) **402** may be used to adjust the time of day such that street lights turn on and off, headlights turn on and off, shadows, glares, and/or sunsets are considered, etc. In some examples, only lights within a threshold distance to the virtual object may be considered to increase efficiency.

Weather may be accounted for by the simulator component(s) **402** (e.g., by the virtual world manager **610**). The weather may be used to update the coefficients of friction for the driving surfaces, and temperature information may be used to update tire interaction with the driving surfaces. Where rain or snow are present, the system **600** may generate meshes to describe where rainwater and snow may accumulate based on the structure of the scene, and the meshes may be employed when rain or snow are present in the simulation.

In some examples, as described herein, at least some of the simulator component(s) **402** may alternatively be included in the vehicle simulator component(s) **420** and/or **406**. For example, the vehicle simulator component(s) **420** and/or the vehicle simulator component(s) **406** may include the virtual sensor manager **612** for managing each of the sensors of the associated virtual object. In addition, one or more of the codecs **614** may be included in the vehicle simulator component(s) **420** and/or the vehicle simulator component(s) **406**. In such examples, the virtual sensor manager **612** may generate sensor data corresponding to a sensor of the virtual object, and the sensor data may be used by sensor emulator **616** of the codec(s) **614** to encode the sensor data according to the sensor data format or type used

by the software stack(s) **116** (e.g., the software stack(s) **116** executing on the vehicle simulator component(s) **420** and/or the vehicle simulator component(s) **406**).

The codec(s) **614** may provide an interface to the software stack(s) **116**. The codec(s) **614** (and/or other codec(s) described herein) may include an encoder/decoder framework. The codec(s) **614** may include CAN steering, throttle requests, and/or may be used to send sensor data to the software stack(s) **116** in SIL and HIL embodiments. The codec(s) **614** may be beneficial to the simulation systems described herein (e.g., **400** and **600**). For example, as data is produced by the re-simulation systems **100** and the simulation systems **400** and **600**, the data may be transmitted to the software stack(s) **116** such that the following standards may be met. The data may be transferred to the software stack(s) **116** such that minimal impact is introduced to the software stack(s) **116** and/or the vehicle hardware **104** (in HIL embodiments). This may result in more accurate simulations as the software stack(s) **116** and/or the vehicle hardware **104** may be operating in an environment that closely resembles deployment in a real-world environment. The data may be transmitted to the software stack(s) **116** such that the simulator and/or re-simulator may be agnostic to the actual hardware configuration of the system under test. This may reduce development overhead due to bugs or separate code paths depending on the simulation configuration. The data may be transmitted to the software stack(s) **116** such that the data may match (e.g., bit-to-bit) the data sent from a physical sensor of a physical vehicle (e.g., the vehicle **102**). The data may be transmitted to efficiently in both SIL and HIL embodiments.

The sensor emulator **616** may emulate at least cameras, LIDAR sensors, and/or RADAR sensors. With respect to LIDAR sensors, some LIDAR sensors report tracked objects. As such, for each frame represented by the virtual sensor data, the simulator component(s) **402** may create a list of all tracked objects (e.g., trees, vehicles, pedestrians, foliage, etc.) within range of the virtual object having the virtual LIDAR sensors, and may cast virtual rays toward the tracked objects. When a significant number of rays strike a tracked object, that object may be added to the report of the LIDAR data. In some examples, the LIDAR sensors may be modeled using simple ray-casting without reflection, adjustable field of view, adjustable noise, and/or adjustable dropouts. LIDAR with moving parts, limited fields of view, and/or variable resolutions may be simulated. For example, the LIDAR sensors may be modeled as solid state LIDAR and/or as Optix-based LIDAR. In examples, using Optix-based LIDAR, the rays may bounce from water, reflective materials, and/or windows. Texture may be assigned to roads, signs, and/or vehicles to model laser reflection at the wavelengths corresponding to the textures. RADAR may be implemented similarly to LIDAR. As described herein, RADAR and/or LIDAR may be simulated using ray-tracing techniques.

In some examples, the vehicle simulator component(s) **406**, **420**, and/or **422** may include a feedback loop with the simulator component(s) **402** (and/or the component(s) that generate the virtual sensor data). The feedback loop may be used to provide information for updating the virtual sensor data capture or generation. For example, for virtual cameras, the feedback loop may be based on sensor feedback, such as changes to exposure responsive to lighting conditions (e.g., increase exposure in dim lighting conditions so that the image data may be processed by the DNNs properly). As another example, for virtual LIDAR sensors, the feedback

loop may be representative of changes to energy level (e.g., to boost energy to produce more useable or accurate LIDAR data).

GNNS sensors (e.g., GPS sensors) may be simulated within the simulation space to generate real-world coordinates. In order to this, noise functions may be used to approximate inaccuracy. As with any virtual sensors described herein, the virtual sensor data may be transmitted to the software stack(s) 116 using the codec(s) 614 to be converted to a bit-to-bit correct signal (e.g., corresponding accurately to the signals generated by the physical sensors of the physical vehicles).

One or more plugin application programming interfaces (APIs) 606 may be used. The plugin APIs 606 may include first-party and/or third-party plugins. For example, third parties may customize the simulation system 600B using their own plugin APIs 606 for providing custom information, such as performance timings, suspension dynamics, tire dynamics, etc.

The plugin APIs 606 may include an ego-dynamics component(s) (not shown) that may receive information from the simulator component(s) 402 including position, velocity, car state, and/or other information, and may provide information to the simulator component(s) 402 including performance timings, suspension dynamics, tire dynamics, and/or other information. For examples, the simulator component(s) 402 may provide CAN throttle, steering, and the driving surface information to the ego-dynamics component(s). In some examples, the ego-dynamics component(s) may include an off-the-shelf vehicle dynamics package (e.g., IPG CAR-MAKER or VIRTUAL TEST DRIVE), while in other examples the ego-dynamics component(s) may be customized and/or received (e.g., from a first-party and/or a third-party).

The plugin APIs 606 may include a key performance indicator (KPI) API. The KPI API may receive CAN data, ground truth, and/or virtual object state information (e.g., from the software stack(s) 116) from the simulator component(s) 402 and may generate and/or provide a report (in real-time) that includes KPI's and/or commands to save state, restore state, and/or apply changes.

Now referring to FIG. 6B, FIG. 6B includes a cloud-based architecture for a simulation system 600B, in accordance with some embodiment of the present disclosure. The simulation system 600B may, at least partly, reside in the cloud and may communicate over one or more networks, such as but not limited to those described herein (e.g., with respect to network 1190 of FIG. 1D), with one or more GPU platforms 624 (e.g., that may include GPUs, CPUs, TPUS, and/or other processor types) and/or one or more HIL platforms 626 (e.g., which may include some or all of the components from the vehicle simulator component(s) 406, described herein).

A simulated environment 628 (e.g., which may be similar to the simulated environment 410 described herein) may be modeled by interconnected components including a simulation engine 630, an AI engine 632, a global illumination (GI) engine 634, an asset data store(s) 636, and/or other components. In some examples, these component(s) may be used to model a simulated environment (e.g., a virtual world) in a virtualized interactive platform (e.g., similar to a massive multiplayer online (MMO) game environment. The simulated environment may further include physics, traffic simulation, weather simulation, and/or other features and simulations for the simulated environment. GI engine 634 may calculate GI once and share the calculation with each of the nodes 618(1)-618(N) and 620(1)-620(N) (e.g.,

the calculation of GI may be view independent). The simulated environment 628 may include an AI universe 622 that provides data to GPU platforms 624 (e.g., GPU servers) that may create renderings for each sensor of the vehicle (e.g., at the virtual sensor/codec(s) 618 for a first virtual object and at the virtual sensor codec(s) 620 for a second virtual object). For example, the GPU platform 624 may receive data about the simulated environment 628 and may create sensor inputs for each of 618(1)-618(N), 620(1)-620(N), and/or virtual sensor/codec pairs corresponding to other virtual objects (depending on the embodiment). In examples where the virtual objects are simulated using HIL objects, the sensor inputs may be provided to the vehicle hardware 104 which may use the software stack(s) 116 to perform one or more operations and/or generate one or more commands, such as those described herein. In some examples, as described herein, the virtual sensor data from each of the virtual sensors may be encoded using a codec prior to being used by (or transmitted to) the software stack(s) 116. In addition, in some examples, each of the sensors may be executed on its own GPU within the GPU platform 624, while in other examples, two or more sensors may share the same GPU within the GPU platform 624.

The one or more operations or commands may be transmitted to the simulation engine 630 which may update the behavior of one or more of the virtual objects based on the operations and/or commands. For example, the simulation engine 630 may use the AI engine 632 to update the behavior of the AI agents as well as the virtual objects in the simulated environment 628. The simulation engine 630 may then update the object data and characteristics (e.g., within the asset data store(s) 636), may update the GI (and/or other aspects such as reflections, shadows, etc.), and then may generate and provide updated sensor inputs to the GPU platform 624. This process may repeat until a simulation is completed.

Now referring to FIG. 7, FIG. 7 includes a data flow diagram illustrating a process 700 for re-simulation or simulation using one or more codecs, in accordance with some embodiments of the present disclosure. The process 700 may include a current state and/or sensor data be transmitted from the simulation and/or re-simulation to one or more codecs 704. At least some of the data (e.g., the sensor data) may then be encoded using the codec(s) 704 and provided to the software stack(s) 706 (e.g., similar to the software stack(s) 116) for a current time slice. The driving commands and new sensor state may then transmitted (e.g., via CAN or V-CAN) to the codec(s) 704 and back to the simulation and/or re-simulation. The driving commands generated originally by the software stack(s) 706 (e.g., by an autonomous driving software stack) may then be passed to ego-object dynamics which may use custom or built-in dynamics to update the object state for the particular type of virtual object being simulated and the updated object state may be passed back to the simulation and/or re-simulation. The simulation system may use the object's state, commands, and/or information, in addition to using traffic AI, pedestrian AI, and/or other features of the simulation platform, to generate or update the simulated environment (e.g., to a current state). The current state may be passed to the KPI framework (e.g., at the same time as the driving commands being passed to the ego-object dynamics 708, in some embodiments), and the KPI framework 710 may monitor and evaluate the current simulation and/or re-simulation. In some examples, the codec(s) 704 may buffer simulation data to increase performance and/or reduce latency of the system.

Now referring to FIG. **8**, FIG. **8** includes a data flow diagram for key performance indicator (KPI) analysis and observation, in accordance with some embodiments of the present disclosure. A KPI evaluation component may evaluate the performance of the virtual object(s) (e.g., vehicles, robots, etc.). Logs **806** may be generated and passed to re-simulator/simulator **804**. The re-simulator/simulator **804** may provide sensor data to the software stack(s) **116** which may be executed using HIL, SIL, or a combination thereof. The KPI evaluation component **802** may use different metrics for each simulation or re-simulation instance. For examples, for re-simulation, KPI evaluation component may provide access to the original re-played CAN data and/or the newly generated CAN data from the software stack(s) **116** (e.g., from HIL or SIL). In some examples, performance could be as simple as testing that the new CAN data does not create a false positive—such as by triggering Automatic Emergency Braking (AEB), or another ADAS functionality. For example, the KPI evaluation component **802** may determine whether the new CAN data triggers a blind spot warning, or a lane departure warning. As a result, the system may help reduce the false positives that plague conventional ADAS systems. The KPI evaluation component **802** may also determine whether the new CAN data fails to trigger a warning that should have been implemented.

In some examples, the KPI evaluation component **802** may also provide for more complex comparisons. For example, the KPI evaluation component **802** may be as complex as running analytics on the two differing CAN streams to find deviations. The KPI evaluation component **802** may compare the new CAN data against the original CAN data, and may evaluate both trajectories to determine which trajectory would best meet the systems safety goals. In some examples, the KPI evaluation component **802** may use one or more methods described in U.S. Provisional Application No. 62/625,351, or U.S. Non-Provisional patent application Ser. No. 16/256,780, each hereby incorporated by reference in its entirety. In other examples, the KPI Evaluation component **802** may use one or of the methods described in U.S. Provisional Application No. 62/628,831, or U.S. Non-Provisional patent application Ser. No. 16/269, 921, each hereby incorporated by reference in its entirety. For example, safety procedures may be determined based on safe time of arrival calculations.

In some examples, the KPI evaluation component **802** may also use the method described in U.S. Provisional Application No. 62/622,538 or U.S. Non-Provisional patent application Ser. No. 16/258,272, hereby incorporated by reference in its entirety, which may be used to detect hazardous driving using machine learning. For example, machine learning and deep neural networks (DNNs) may be used for redundancy and for path checking e.g., for a rationality checker as part of functional safety for autonomous driving. These techniques may be extended for use with the KPI evaluation component **802** to evaluate the performance of the system.

The KPI Evaluation component may also use additional approaches to assess the performance of the system. For example, the KPI evaluation component **802** may consider whether the time to arrival (TTA) in the path of the cross-traffic is less than a threshold time—e.g. two seconds. The threshold may vary depending on the speed of the vehicle, road conditions, weather, traffic, and/or other variables. For example, the threshold duration may be two seconds for speeds up to twenty MPH, and one second for any greater speed. Alternatively, the threshold duration may be reduced or capped whenever the system detects hazardous road conditions such as wet roads, ice, or snow. In some examples, hazardous road conditions may be detected by a DNN trained to detect such conditions.

With respect to simulation, the KPI evaluation component may include an API, as described herein. The KPI evaluation component **802** may include additional inputs and/or provide more functionality. For example, the simulator may be able to share the "ground truth" for the scene, and may be able to determine the capability of the virtual object with respect to avoiding collisions, staying-in-lane, and/or performing other behaviors. For examples, the KPI evaluation component **802** may be more than a passive witness to the experiment, and may include an API to save the state of any ongoing simulation, change state or trigger behaviors, and continue with those changes. This may allow the KPI evaluation component to not only evaluate the car performance but to try to explore the space of potential dangerous scenarios.

Now referring to FIGS. **9** and **10**, each block of methods **900** and **1000**, described herein, comprises a computing process that may be performed using any combination of hardware, firmware, and/or software. For instance, various functions may be carried out by a processor executing instructions stored in memory. The methods may also be embodied as computer-usable instructions stored on computer storage media. The methods may be provided by a standalone application, a service or hosted service (standalone or in combination with another hosted service), or a plug-in to another product, to name a few. In addition, methods **900** and **1000** are described, by way of example, with respect to the re-simulation system **100** of FIG. **1**, the simulation system **400** of FIGS. **4A-4C**, and the simulation system **600** of FIGS. **6A-6B**. However, these methods may additionally or alternatively be executed by any one system, or any combination of systems, including, but not limited to, those described herein.

FIG. **9** is a flow diagram showing a method **900** for controlling a virtual object in a simulated environment, in accordance with some embodiments of the present disclosure. The method **900**, at block B**902**, includes receiving simulation data representative of a simulated environment from a simulation host device. For example, the vehicle simulator component(s) **406**, **420**, and/or **422** may receive, from the simulator component(s) **402**, simulation data representative of the simulated environment **410**. In some examples, the simulation data received may be the simulation data corresponding to the sensors of the virtual object hosted by the vehicle simulator component(s).

The method **900**, at block B**904**, includes generating virtual sensor data for each of a dynamically configurable number of virtual sensors. For example, the vehicle simulator component(s) **406**, **420**, and/or **422** may generate virtual sensor data using the simulation data for each of the virtual sensors of the vehicle. The virtual sensor data may be representative of the simulated environment **410** as perceived by at least one virtual sensor of a dynamically configurable number of virtual sensors of a virtual object within the simulated environment **410** (e.g., sensor data of a field of view of a virtual camera(s), sensor data of an orientation of the virtual vehicle using virtual IMU sensors, etc.). The number of virtual sensors used may be dynamically configurable such that one sensor may be used in a first simulation, five in another, ten in another, etc. In some examples, the dynamic configuration may be determined based on vehicle types (e.g., a first vehicle of year X, make Y, model Z may include 20 sensors, while a second vehicle of year A, make B, model C may include 30 sensors). In such

examples, the simulation system **400, 600** may be dynamically configurable to generate virtual sensor data for each of the virtual sensors of each, or any vehicle in the simulated environment. In addition, any number of different virtual objects may be simulated within the simulated environment at any one time. As such, for each of a first virtual object (e.g., executing on a first set of vehicle simulator component(s) **406**), a second virtual object (e.g., executing on a second set of vehicle simulator component(s) **420**), and/or any other virtual objects, a same or different number of virtual sensors and/or type of virtual sensors may generate virtual sensor data. The virtual sensor data for each virtual sensor may be representative of any other virtual objects as perceived by the respective virtual sensor. As such, the simulation system **400, 600** (e.g., using the DSM **424**) may generate virtual sensor data for each of the virtual sensors that reflects the simulation state of the simulated environment with respect to each other virtual object. In this way, the simulation system is scalable and configurable to any number of virtual objects each having any number of virtual sensors that may each be processed in real-time.

The method **900**, at block B**906**, includes encoding the virtual sensor data. For example, the virtual sensor data may be encoded using one or more codecs (e.g., codec(s) **614**) to generate encoded sensor data. In some examples, the virtual sensor data may be encoded to a format that is familiar to the software stack(s) **116** of the virtual object.

The method **900**, at block B**908**, includes computing, by one or more machine learning models, at least one output. For example, one or more DNNs of the software stack(s) **116** may uses the encoded sensor data to generate one or more outputs (e.g., objects detections, controls, actuations, path plans, guidance, etc.). In some examples, as described herein, the software stack(s) **116** may be executed on the vehicle hardware **104** (e.g., for HIL objects), so the software and hardware used in the physical vehicle (e.g., physical vehicle) may be used in the simulation system **400** and **600** to more accurately produce results consistent with real-world deployment.

The method **900**, at block B**910**, includes transmitting, to the simulation host device, a signal. For example, the output (or data representative thereof) may be transmitted in a signal to the simulation component(s) **402** to update the global simulation, and thus the simulation environment.

FIG. **10** is a flow diagram showing a method **1000** for controlling a virtual object in a simulated environment using machine learning models trained on physical sensor data, in accordance with some embodiments of the present disclosure. The method **1000**, at block B**1002**, includes receiving physical sensor data generated by a physical sensor. For example, the vehicle(s) **102** (e.g., a physical or physical vehicle(s)) may generate physical sensor data and the re-simulation and/or simulation system may receive the physical sensor data.

The method **1000**, at block B**1004**, includes training a machine learning model using the physical sensor data. For example, one or more DNNs that may be used in the software stack(s) **116** may be trained using the physical sensor data. Once trained, the DNN's may be considered trained DNNs.

The method **1000**, at block B**1006**, includes receiving virtual sensor data generated by a virtual sensor. For example, the vehicle simulator component(s) **406, 420**, and/or **422** may generate virtual sensor data using one or more virtual sensors and/or one or more codecs.

The method **1000**, at block B**1008**, includes applying the virtual sensor data to a trained machine learning model. For

example, the virtual sensor data—which may be in the same format as the physical sensor data that was used to train the machine learning model—may be applied to the trained machine learning model.

The method **1000**, at block B**1010**, includes computing an output by the trained machine learning model. For example, the trained DNN may compute one or more outputs using the virtual sensor data. As described herein, the virtual sensor data may be encoded prior to use by the trained DNN.

The method **1000**, at block B**1020**, includes controlling a virtual object within a simulated environment based at least in part on the output. For example, the virtual object (e.g., virtual vehicle) may be controlled within the simulated environment based at least in part on the output. In other examples, the outputs may be used for control. For example, the outputs may be object detection, lane detection, drivable free-space detection, safety procedure determination, etc. In any example, the outputs may be tested using one or more KPI's to determine the accuracy and effectiveness of the trained DNNs in any of a number of scenarios and environments. As such, where the trained DNNs suffer, fine-tuning may be executed to improve, validate, and verify the DNNs prior to deployment of the DNNs in real-world, physical vehicles (e.g., the vehicle **102**).

Example Autonomous Vehicle

FIG. **11A** is an illustration of an example autonomous vehicle **102**, in accordance with some embodiments of the present disclosure. The autonomous vehicle **102** (alternatively referred to herein as the "vehicle **102**") may include a passenger vehicle, such as a car, a truck, a bus, and/or another type of vehicle that accommodates one or more passengers. Autonomous vehicles are generally described in terms of automation levels, defined by the National Highway Traffic Safety Administration (NHTSA), a division of the US Department of Transportation, and the Society of Automotive Engineers (SAE) "Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles" (Standard No. J3016-201806, published on Jun. 15, 2018, Standard No. J3016-201609, published on Sep. 30, 2016, and previous and future versions of this standard). The vehicle **102** may be capable of functionality in accordance with one or more of Level 3-Level 5 of the autonomous driving levels. For example, the vehicle **102** may be capable of conditional automation (Level 3), high automation (Level 4), and/or full automation (Level 5), depending on the embodiment.

The vehicle **102** may include components such as a chassis, a vehicle body, wheels (e.g., 2, 4, 6, 8, 18, etc.), tires, axles, and other components of a vehicle. The vehicle **102** may include a propulsion system **1150**, such as an internal combustion engine, hybrid electric power plant, an all-electric engine, and/or another propulsion system type. The propulsion system **1150** may be connected to a drive train of the vehicle **102**, which may include a transmission, to enable the propulsion of the vehicle **102**. The propulsion system **1150** may be controlled in response to receiving signals from the throttle/accelerator **1152**.

A steering system **1154**, which may include a steering wheel, may be used to steer the vehicle **102** (e.g., along a desired path or route) when the propulsion system **1150** is operating (e.g., when the vehicle is in motion). The steering system **1154** may receive signals from a steering actuator **1156**. The steering wheel may be optional for full automation (Level 5) functionality.

The brake sensor system **1146** may be used to operate the vehicle brakes in response to receiving signals from the brake actuators **1148** and/or brake sensors.

Controller(s) **1136**, which may include one or more system on chips (SoCs) **1104** (FIG. **11C**) and/or GPU(s), may provide signals (e.g., representative of commands) to one or more components and/or systems of the vehicle **102**. For example, the controller(s) may send signals to operate the vehicle brakes via one or more brake actuators **1148**, to operate the steering system **1154** via one or more steering actuators **1156**, to operate the propulsion system **1150** via one or more throttle/accelerators **1152**. The controller(s) **1136** may include one or more onboard (e.g., integrated) computing devices (e.g., supercomputers) that process sensor signals, and output operation commands (e.g., signals representing commands) to enable autonomous driving and/or to assist a human driver in driving the vehicle **102**. The controller(s) **1136** may include a first controller **1136** for autonomous driving functions, a second controller **1136** for functional safety functions, a third controller **1136** for artificial intelligence functionality (e.g., computer vision), a fourth controller **1136** for infotainment functionality, a fifth controller **1136** for redundancy in emergency conditions, and/or other controllers. In some examples, a single controller **1136** may handle two or more of the above functionalities, two or more controllers **1136** may handle a single functionality, and/or any combination thereof.

The controller(s) **1136** may provide the signals for controlling one or more components and/or systems of the vehicle **102** in response to sensor data received from one or more sensors (e.g., sensor inputs). The sensor data may be received from, for example and without limitation, global navigation satellite systems sensor(s) **1158** (e.g., Global Positioning System sensor(s)), RADAR sensor(s) **1160**, ultrasonic sensor(s) **1162**, LIDAR sensor(s) **1164**, inertial measurement unit (IMU) sensor(s) **1166** (e.g., accelerometer(s), gyroscope(s), magnetic compass(es), magnetometer(s), etc.), microphone(s) **1196**, stereo camera(s) **1168**, wide-view camera(s) **1170** (e.g., fisheye cameras), infrared camera(s) **1172**, surround camera(s) **1174** (e.g., 360 degree cameras), long-range and/or mid-range camera(s) **1198**, speed sensor(s) **1144** (e.g., for measuring the speed of the vehicle **102**), vibration sensor(s) **1142**, steering sensor(s) **1140**, brake sensor(s) (e.g., as part of the brake sensor system **1146**), and/or other sensor types.

One or more of the controller(s) **1136** may receive inputs (e.g., represented by input data) from an instrument cluster **1132** of the vehicle **102** and provide outputs (e.g., represented by output data, display data, etc.) via a human-machine interface (HMI) display **1134**, an audible annunciator, a loudspeaker, and/or via other components of the vehicle **102**. The outputs may include information such as vehicle velocity, speed, time, map data (e.g., the HD map **1122** of FIG. **11C**), location data (e.g., the vehicle's **102** location, such as on a map), direction, location of other vehicles (e.g., an occupancy grid), information about objects and status of objects as perceived by the controller(s) **1136**, etc. For example, the HMI display **1134** may display information about the presence of one or more objects (e.g., a street sign, caution sign, traffic light changing, etc.), and/or information about driving maneuvers the vehicle has made, is making, or will make (e.g., changing lanes now, taking exit **34B** in two miles, etc.).

The vehicle **102** further includes a network interface **1124** which may use one or more wireless antenna(s) **1126** and/or modem(s) to communicate over one or more networks. For example, the network interface **1124** may be capable of

communication over LTE, WCDMA, UMTS, GSM, CDMA2000, etc. The wireless antenna(s) **1126** may also enable communication between objects in the environment (e.g., vehicles, mobile devices, etc.), using local area network(s), such as Bluetooth, Bluetooth LE, Z-Wave, ZigBee, etc., and/or low power wide-area network(s) (LPWANs), such as LoRaWAN, SigFox, etc.

FIG. **11B** is an example of camera locations and fields of view for the example autonomous vehicle **102** of FIG. **11A**, in accordance with some embodiments of the present disclosure. The cameras and respective fields of view are one example embodiment and are not intended to be limiting. For example, additional and/or alternative cameras may be included and/or the cameras may be located at different locations on the vehicle **102**.

The camera types for the cameras may include, but are not limited to, digital cameras that may be adapted for use with the components and/or systems of the vehicle **102**. The camera(s) may operate at automotive safety integrity level (ASIL) B and/or at another ASIL. The camera types may be capable of any image capture rate, such as 60 frames per second (fps), 1120 fps, 240 fps, etc., depending on the embodiment. The cameras may be capable of using rolling shutters, global shutters, another type of shutter, or a combination thereof. In some examples, the color filter array may include a red clear clear clear (RCCC) color filter array, a red clear clear blue (RCCB) color filter array, a red blue green clear (RBGC) color filter array, a Foveon X3 color filter array, a Bayer sensors (RGGB) color filter array, a monochrome sensor color filter array, and/or another type of color filter array. In some embodiments, clear pixel cameras, such as cameras with an RCCC, an RCCB, and/or an RBGC color filter array, may be used in an effort to increase light sensitivity.

In some examples, one or more of the camera(s) may be used to perform advanced driver assistance systems (ADAS) functions (e.g., as part of a redundant or fail-safe design). For example, a Multi-Function Mono Camera may be installed to provide functions including lane departure warning, traffic sign assist and intelligent headlamp control. One or more of the camera(s) (e.g., all of the cameras) may record and provide image data (e.g., video) simultaneously.

One or more of the cameras may be mounted in a mounting assembly, such as a custom designed (3-D printed) assembly, in order to cut out stray light and reflections from within the car (e.g., reflections from the dashboard reflected in the windshield mirrors) which may interfere with the camera's image data capture abilities. With reference to wing-mirror mounting assemblies, the wing-mirror assemblies may be custom 3-D printed so that the camera mounting plate matches the shape of the wing-mirror. In some examples, the camera(s) may be integrated into the wing-mirror. For side-view cameras, the camera(s) may also be integrated within the four pillars at each corner of the cabin.

Cameras with a field of view that include portions of the environment in front of the vehicle **102** (e.g., front-facing cameras) may be used for surround view, to help identify forward facing paths and obstacles, as well aid in, with the help of one or more controllers **1136** and/or control SoCs, providing information critical to generating an occupancy grid and/or determining the preferred vehicle paths. Front-facing cameras may be used to perform many of the same ADAS functions as LIDAR, including emergency braking, pedestrian detection, and collision avoidance. Front-facing cameras may also be used for ADAS functions and systems

including Lane Departure Warnings ("LDW"), Autonomous Cruise Control ("ACC"), and/or other functions such as traffic sign recognition.

A variety of cameras may be used in a front-facing configuration, including, for example, a monocular camera platform that includes a CMOS (complementary metal oxide semiconductor) color imager. Another example may be a wide-view camera(s) 1170 that may be used to perceive objects coming into view from the periphery (e.g., pedestrians, crossing traffic or bicycles). Although only one wide-view camera is illustrated in FIG. 11B, there may any number of wide-view cameras 1170 on the vehicle 102. In addition, long-range camera(s) 1198 (e.g., a long-view stereo camera pair) may be used for depth-based object detection, especially for objects for which a neural network has not yet been trained. The long-range camera(s) 1198 may also be used for object detection and classification, as well as basic object tracking.

One or more stereo cameras 1168 may also be included in a front-facing configuration. The stereo camera(s) 1168 may include an integrated control unit comprising a scalable processing unit, which may provide a programmable logic (FPGA) and a multi-core micro-processor with an integrated CAN or Ethernet interface on a single chip. Such a unit may be used to generate a 3-D map of the vehicle's environment, including a distance estimate for all the points in the image. An alternative stereo camera(s) 1168 may include a compact stereo vision sensor(s) that may include two camera lenses (one each on the left and right) and an image processing chip that may measure the distance from the vehicle to the target object and use the generated information (e.g., metadata) to activate the autonomous emergency braking and lane departure warning functions. Other types of stereo camera(s) 1168 may be used in addition to, or alternatively from, those described herein.

Cameras with a field of view that include portions of the environment to the side of the vehicle 102 (e.g., side-view cameras) may be used for surround view, providing information used to create and update the occupancy grid, as well as to generate side impact collision warnings. For example, surround camera(s) 1174 (e.g., four surround cameras 1174 as illustrated in FIG. 11B) may be positioned to on the vehicle 102. The surround camera(s) 1174 may include wide-view camera(s) 1170, fisheye camera(s), 360 degree camera(s), and/or the like. Four example, four fisheye cameras may be positioned on the vehicle's front, rear, and sides. In an alternative arrangement, the vehicle may use three surround camera(s) 1174 (e.g., left, right, and rear), and may leverage one or more other camera(s) (e.g., a forward-facing camera) as a fourth surround view camera.

Cameras with a field of view that include portions of the environment to the rear of the vehicle 102 (e.g., rear-view cameras) may be used for park assistance, surround view, rear collision warnings, and creating and updating the occupancy grid. A wide variety of cameras may be used including, but not limited to, cameras that are also suitable as a front-facing camera(s) (e.g., long-range and/or mid-range camera(s) 1198, stereo camera(s) 1168), infrared camera(s) 1172, etc.), as described herein.

FIG. 11C is a block diagram of an example system architecture for the example autonomous vehicle 102 of FIG. 11A, in accordance with some embodiments of the present disclosure. It should be understood that this and other arrangements described herein are set forth only as examples. Other arrangements and elements (e.g., machines, interfaces, functions, orders, groupings of functions, etc.) may be used in addition to or instead of those shown, and

some elements may be omitted altogether. Further, many of the elements described herein are functional entities that may be implemented as discrete or distributed components or in conjunction with other components, and in any suitable combination and location. Various functions described herein as being performed by entities may be carried out by hardware, firmware, and/or software. For instance, various functions may be carried out by a processor executing instructions stored in memory.

Each of the components, features, and systems of the vehicle 102 in FIG. 11C are illustrated as being connected via bus 1102. The bus 1102 may include a Controller Area Network (CAN) data interface (alternatively referred to herein as a "CAN bus"). A CAN may be a network inside the vehicle 102 used to aid in control of various features and functionality of the vehicle 102, such as actuation of brakes, acceleration, braking, steering, windshield wipers, etc. A CAN bus may be configured to have dozens or even hundreds of nodes, each with its own unique identifier (e.g., a CAN ID). The CAN bus may be read to find steering wheel angle, ground speed, engine revolutions per minute (RPMs), button positions, and/or other vehicle status indicators. The CAN bus may be ASIL B compliant.

Although the bus 1102 is described herein as being a CAN bus, this is not intended to be limiting. For example, in addition to, or alternatively from, the CAN bus, FlexRay and/or Ethernet may be used. Additionally, although a single line is used to represent the bus 1102, this is not intended to be limiting. For example, there may be any number of busses 1102, which may include one or more CAN busses, one or more FlexRay busses, one or more Ethernet busses, and/or one or more other types of busses using a different protocol. In some examples, two or more busses 1102 may be used to perform different functions, and/or may be used for redundancy. For example, a first bus 1102 may be used for collision avoidance functionality and a second bus 1102 may be used for actuation control. In any example, each bus 1102 may communicate with any of the components of the vehicle 102, and two or more busses 1102 may communicate with the same components. In some examples, each SoC 1104, each controller 1136, and/or each computer within the vehicle may have access to the same input data (e.g., inputs from sensors of the vehicle 102), and may be connected to a common bus, such the CAN bus.

The vehicle 102 may include one or more controller(s) 1136, such as those described herein with respect to FIG. 11A. The controller(s) 1136 may be used for a variety of functions. The controller(s) 1136 may be coupled to any of the various other components and systems of the vehicle 102, and may be used for control of the vehicle 102, artificial intelligence of the vehicle 102, infotainment for the vehicle 102, and/or the like.

The vehicle 102 may include a system(s) on a chip (SoC) 1104. The SoC 1104 may include CPU(s) 1106, GPU(s) 1108, processor(s) 1110, cache(s) 1112, accelerator(s) 1114, data store(s) 1116, and/or other components and features not illustrated. The SoC(s) 1104 may be used to control the vehicle 102 in a variety of platforms and systems. For example, the SoC(s) 1104 may be combined in a system (e.g., the system of the vehicle 102) with an HD map 1122 which may obtain map refreshes and/or updates via a network interface 1124 from one or more servers (e.g., server(s) 1178 of FIG. 11D).

The CPU(s) 1106 may include a CPU cluster or CPU complex (alternatively referred to herein as a "CCPLEX"). The CPU(s) 1106 may include multiple cores and/or L2 caches. For example, in some embodiments, the CPU(s)

1106 may include eight cores in a coherent multi-processor configuration. In some embodiments, the CPU(s) **1106** may include four dual-core clusters where each cluster has a dedicated L2 cache (e.g., a 2 MB L2 cache). The CPU(s) **1106** (e.g., the CCPLEX) may be configured to support simultaneous cluster operation enabling any combination of the clusters of the CPU(s) **1106** to be active at any given time.

The CPU(s) **1106** may implement power management capabilities that include one or more of the following features: individual hardware blocks may be clock-gated automatically when idle to save dynamic power; each core clock may be gated when the core is not actively executing instructions due to execution of WFI/WFE instructions; each core may be independently power-gated; each core cluster may be independently clock-gated when all cores are clock-gated or power-gated; and/or each core cluster may be independently power-gated when all cores are power-gated. The CPU(s) **1106** may further implement an enhanced algorithm for managing power states, where allowed power states and expected wakeup times are specified, and the hardware/microcode determines the best power state to enter for the core, cluster, and CCPLEX. The processing cores may support simplified power state entry sequences in software with the work offloaded to microcode.

The GPU(s) **1108** may include an integrated GPU (alternatively referred to herein as an "iGPU"). The GPU(s) **1108** may be programmable and may be efficient for parallel workloads. The GPU(s) **1108**, in some examples, may use an enhanced tensor instruction set. The GPU(s) **1108** may include one or more streaming microprocessors, where each streaming microprocessor may include an L1 cache (e.g., an L1 cache with at least 96 KB storage capacity), and two or more of the streaming microprocessors may share an L2 cache (e.g., an L2 cache with a 512 KB storage capacity). In some embodiments, the GPU(s) **1108** may include at least eight streaming microprocessors. The GPU(s) **1108** may use compute application programming interface(s) (API(s)). In addition, the GPU(s) **1108** may use one or more parallel computing platforms and/or programming models (e.g., NVIDIA's CUDA).

The GPU(s) **1108** may be power-optimized for best performance in automotive and embedded use cases. For example, the GPU(s) **1108** may be fabricated on a Fin field-effect transistor (FinFET). However, this is not intended to be limiting and the GPU(s) **1108** may be fabricated using other semiconductor manufacturing processes. Each streaming microprocessor may incorporate a number of mixed-precision processing cores partitioned into multiple blocks. For example, and without limitation, 64 PF32 cores and 32 PF64 cores may be partitioned into four processing blocks. In such an example, each processing block may be allocated 16 FP32 cores, 8 FP64 cores, 16 INT32 cores, two mixed-precision NVIDIA TENSOR COREs for deep learning matrix arithmetic, an L0 instruction cache, a warp scheduler, a dispatch unit, and/or a 64 KB register file. In addition, the streaming microprocessors may include independent parallel integer and floating-point data paths to provide for efficient execution of workloads with a mix of computation and addressing calculations. The streaming microprocessors may include independent thread scheduling capability to enable finer-grain synchronization and cooperation between parallel threads. The streaming microprocessors may include a combined L1 data cache and shared memory unit in order to improve performance while simplifying programming.

The GPU(s) **1108** may include a high bandwidth memory (HBM) and/or a 16 GB HBM2 memory subsystem to provide, in some examples, about 900 GB/second peak memory bandwidth. In some examples, in addition to, or alternatively from, the HBM memory, a synchronous graphics random-access memory (SGRAM) may be used, such as a graphics double data rate type five synchronous random-access memory (GDDR5).

The GPU(s) **1108** may include unified memory technology including access counters to allow for more accurate migration of memory pages to the processor that accesses them most frequently, thereby improving efficiency for memory ranges shared between processors. In some examples, address translation services (ATS) support may be used to allow the GPU(s) **1108** to access the CPU(s) **1106** page tables directly. In such examples, when the GPU(s) **1108** memory management unit (MMU) experiences a miss, an address translation request may be transmitted to the CPU(s) **1106**. In response, the CPU(s) **1106** may look in its page tables for the virtual-to-physical mapping for the address and transmits the translation back to the GPU(s) **1108**. As such, unified memory technology may allow a single unified virtual address space for memory of both the CPU(s) **1106** and the GPU(s) **1108**, thereby simplifying the GPU(s) **1108** programming and porting of applications to the GPU(s) **1108**.

In addition, the GPU(s) **1108** may include an access counter that may keep track of the frequency of access of the GPU(s) **1108** to memory of other processors. The access counter may help ensure that memory pages are moved to the physical memory of the processor that is accessing the pages most frequently.

The SoC(s) **1104** may include any number of cache(s) **1112**, including those described herein. For example, the cache(s) **1112** may include an L3 cache that is available to both the CPU(s) **1106** and the GPU(s) **1108** (e.g., that is connected both the CPU(s) **1106** and the GPU(s) **1108**). The cache(s) **1112** may include a write-back cache that may keep track of states of lines, such as by using a cache coherence protocol (e.g., MEI, MESI, MSI, etc.). The L3 cache may include 4 MB or more, depending on the embodiment, although smaller cache sizes may be used.

The SoC(s) **1104** may include one or more accelerators **1114** (e.g., hardware accelerators, software accelerators, or a combination thereof). For example, the SoC(s) **1104** may include a hardware acceleration cluster that may include optimized hardware accelerators and/or large on-chip memory. The large on-chip memory (e.g., 4 MB of SRAM), may enable the hardware acceleration cluster to accelerate neural networks and other calculations. The hardware acceleration cluster may be used to complement the GPU(s) **1108** and to off-load some of the tasks of the GPU(s) **1108** (e.g., to free up more cycles of the GPU(s) **1108** for performing other tasks). As an example, the accelerator(s) **1114** may be used for targeted workloads (e.g., perception, convolutional neural networks (CNNs), etc.) that are stable enough to be amenable to acceleration. The term "CNN," as used herein, may include all types of CNNs, including region-based or regional convolutional neural networks (RCNNs) and Fast RCNNs (e.g., as used for object detection).

The accelerator(s) **1114** (e.g., the hardware acceleration cluster) may include a deep learning accelerator(s) (DLA). The DLA(s) may include one or more Tensor processing units (TPUs) that may be configured to provide an additional ten trillion operations per second for deep learning applications and inferencing. The TPUs may be accelerators configured to, and optimized for, performing image processing

functions (e.g., for CNNs, RCNNs, etc.). The DLA(s) may further be optimized for a specific set of neural network types and floating point operations, as well as inferencing. The design of the DLA(s) may provide more performance per millimeter than a general-purpose GPU, and vastly exceeds the performance of a CPU. The TPU(s) may perform several functions, including a single-instance convolution function, supporting, for example, INT8, INT16, and FP16 data types for both features and weights, as well as post-processor functions.

The DLA(s) may quickly and efficiently execute neural networks, especially CNNs, on processed or unprocessed data for any of a variety of functions, including, for example and without limitation: a CNN for object identification and detection using data from camera sensors; a CNN for distance estimation using data from camera sensors; a CNN for emergency vehicle detection and identification and detection using data from microphones; a CNN for facial recognition and vehicle owner identification using data from camera sensors; and/or a CNN for security and/or safety related events.

The DLA(s) may perform any function of the GPU(s) 1108, and by using an inference accelerator, for example, a designer may target either the DLA(s) or the GPU(s) 1108 for any function. For example, the designer may focus processing of CNNs and floating point operations on the DLA(s) and leave other functions to the GPU(s) 1108 and/or other accelerator(s) 1114.

The accelerator(s) 1114 (e.g., the hardware acceleration cluster) may include a programmable vision accelerator(s) (PVA), which may alternatively be referred to herein as a computer vision accelerator. The PVA(s) may be designed and configured to accelerate computer vision algorithms for the advanced driver assistance systems (ADAS), autonomous driving, and/or augmented reality (AR) and/or virtual reality (VR) applications. The PVA(s) may provide a balance between performance and flexibility. For example, each PVA(s) may include, for example and without limitation, any number of reduced instruction set computer (RISC) cores, direct memory access (DMA), and/or any number of vector processors.

The RISC cores may interact with image sensors (e.g., the image sensors of any of the cameras described herein), image signal processor(s), and/or the like. Each of the RISC cores may include any amount of memory. The RISC cores may use any of a number of protocols, depending on the embodiment. In some examples, the RISC cores may execute a real-time operating system (RTOS). The RISC cores may be implemented using one or more integrated circuit devices, application specific integrated circuits (ASICs), and/or memory devices. For example, the RISC cores may include an instruction cache and/or a tightly coupled RAM.

The DMA may enable components of the PVA(s) to access the system memory independently of the CPU(s) 1106. The DMA may support any number of features used to provide optimization to the PVA including, but not limited to, supporting multi-dimensional addressing and/or circular addressing. In some examples, the DMA may support up to six or more dimensions of addressing, which may include block width, block height, block depth, horizontal block stepping, vertical block stepping, and/or depth stepping.

The vector processors may be programmable processors that may be designed to efficiently and flexibly execute programming for computer vision algorithms and provide signal processing capabilities. In some examples, the PVA may include a PVA core and two vector processing subsys-

tem partitions. The PVA core may include a processor subsystem, DMA engine(s) (e.g., two DMA engines), and/or other peripherals. The vector processing subsystem may operate as the primary processing engine of the PVA, and may include a vector processing unit (VPU), an instruction cache, and/or vector memory (e.g., VMEM). A VPU core may include a digital signal processor such as, for example, a single instruction, multiple data (SIMD), very long instruction word (VLIW) digital signal processor. The combination of the SIMD and VLIW may enhance throughput and speed.

Each of the vector processors may include an instruction cache and may be coupled to dedicated memory. As a result, in some examples, each of the vector processors may be configured to execute independently of the other vector processors. In other examples, the vector processors that are included in a particular PVA may be configured to employ data parallelism. For example, in some embodiments, the plurality of vector processors included in a single PVA may execute the same computer vision algorithm, but on different regions of an image. In other examples, the vector processors included in a particular PVA may simultaneously execute different computer vision algorithms, on the same image, or even execute different algorithms on sequential images or portions of an image. Among other things, any number of PVAs may be included in the hardware acceleration cluster and any number of vector processors may be included in each of the PVAs. In addition, the PVA(s) may include additional error correcting code (ECC) memory, to enhance overall system safety.

The accelerator(s) 1114 (e.g., the hardware acceleration cluster) may include a computer vision network on-chip and SRAM, for providing a high-bandwidth, low latency SRAM for the accelerator(s) 1114. In some examples, the on-chip memory may include at least 4 MB SRAM, consisting of, for example and without limitation, eight field-configurable memory blocks, that may be accessible by both the PVA and the DLA. Each pair of memory blocks may include an advanced peripheral bus (APB) interface, configuration circuitry, a controller, and a multiplexer. Any type of memory may be used. The PVA and DLA may access the memory via a backbone that provides the PVA and DLA with high-speed access to memory. The backbone may include a computer vision network on-chip that interconnects the PVA and the DLA to the memory (e.g., using the APB).

The computer vision network on-chip may include an interface that determines, before transmission of any control signal/address/data, that both the PVA and the DLA provide ready and valid signals. Such an interface may provide for separate phases and separate channels for transmitting control signals/addresses/data, as well as burst-type communications for continuous data transfer. This type of interface may comply with ISO 26262 or IEC 61508 standards, although other standards and protocols may be used.

In some examples, the SoC(s) 1104 may include a real-time ray-tracing hardware accelerator, such as described in U.S. patent application Ser. No. 16/101,232, filed on Aug. 10, 2018. The real-time ray-tracing hardware accelerator may be used to quickly and efficiently determine the positions and extents of objects (e.g., within a world model), to generate real0time visualization simulations, for RADAR signal interpretation, for sound propagation synthesis and/or analysis, for simulation of SONAR systems, for general wave propagation simulation, for comparison to LIDAR data for purposes of localization and/or other functions, and/or for other uses.

The accelerator(s) 1114 (e.g., the hardware accelerator cluster) have a wide array of uses for autonomous driving.

The PVA may be a programmable vision accelerator that may be used for key processing stages in ADAS and autonomous vehicles. The PVA's capabilities are a good match for algorithmic domains needing predictable processing, at low power and low latency. In other words, the PVA performs well on semi-dense or dense regular computation, even on small data sets, which need predictable run-times with low latency and low power. Thus, in the context of platforms for autonomous vehicles, the PVAs are designed to run classic computer vision algorithms, as they are efficient at object detection and operating on integer math.

For example, according to one embodiment of the technology, the PVA is used to perform computer stereo vision. A semi-global matching-based algorithm may be used in some examples, although this is not intended to be limiting. Many applications for Level 3-5 autonomous driving require motion estimation/stereo matching on-the-fly (e.g., structure from motion, pedestrian recognition, lane detection, etc.). The PVA may perform computer stereo vision function on inputs from two monocular cameras.

In some examples, the PVA may be used to perform dense optical flow. According to process raw RADAR data (e.g., using a 4D Fast Fourier Transform) to provide Processed RADAR. In other examples, the PVA is used for time of flight depth processing, by processing raw time of flight data to provide processed time of flight data, for example.

The DLA may be used to run any type of network to enhance control and driving safety, including for example, a neural network that outputs a measure of confidence for each object detection. Such a confidence value may be interpreted as a probability, or as providing a relative "weight" of each detection compared to other detections. This confidence value enables the system to make further decisions regarding which detections should be considered as true positive detections rather than false positive detections. For example, the system may set a threshold value for the confidence and consider only the detections exceeding the threshold value as true positive detections. In an automatic emergency braking (AEB) system, false positive detections would cause the vehicle to automatically perform emergency braking, which is obviously undesirable. Therefore, only the most confident detections should be considered as triggers for AEB. The DLA may run a neural network for regressing the confidence value. The neural network may take as its input at least some subset of parameters, such as bounding box dimensions, ground plane estimate obtained (e.g. from another subsystem), inertial measurement unit (IMU) sensor 1166 output that correlates with the vehicle 102 orientation, distance, 3D location estimates of the object obtained from the neural network and/or other sensors (e.g., LIDAR sensor(s) 1164 or RADAR sensor(s) 1160), among others.

The SoC(s) 1104 may include data store(s) 1116 (e.g., memory). The data store(s) 1116 may be on-chip memory of the SoC(s) 1104, which may store neural networks to be executed on the GPU and/or the DLA. In some examples, the data store(s) 1116 may be large enough in capacity to store multiple instances of neural networks for redundancy and safety. The data store(s) 1112 may comprise L2 or L3 cache(s) 1112. Reference to the data store(s) 1116 may include reference to the memory associated with the PVA, DLA, and/or other accelerator(s) 1114, as described herein.

The SoC(s) 1104 may include one or more processor(s) 1110 (e.g., embedded processors). The processor(s) 1110 may include a boot and power management processor that may be a dedicated processor and subsystem to handle boot power and management functions and related security enforcement. The boot and power management processor

may be a part of the SoC(s) 1104 boot sequence and may provide runtime power management services. The boot power and management processor may provide clock and voltage programming, assistance in system low power state transitions, management of SoC(s) 1104 thermals and temperature sensors, and/or management of the SoC(s) 1104 power states. Each temperature sensor may be implemented as a ring-oscillator whose output frequency is proportional to temperature, and the SoC(s) 1104 may use the ring-oscillators to detect temperatures of the CPU(s) 1106, GPU(s) 1108, and/or accelerator(s) 1114. If temperatures are determined to exceed a threshold, the boot and power management processor may enter a temperature fault routine and put the SoC(s) 1104 into a lower power state and/or put the vehicle 102 into a chauffeur to safe stop mode (e.g., bring the vehicle 102 to a safe stop).

The processor(s) 1110 may further include a set of embedded processors that may serve as an audio processing engine. The audio processing engine may be an audio subsystem that enables full hardware support for multi-channel audio over multiple interfaces, and a broad and flexible range of audio I/O interfaces. In some examples, the audio processing engine is a dedicated processor core with a digital signal processor with dedicated RAM.

The processor(s) 1110 may further include an always on processor engine that may provide necessary hardware features to support low power sensor management and wake use cases. The always on processor engine may include a processor core, a tightly coupled RAM, supporting peripherals (e.g., timers and interrupt controllers), various I/O controller peripherals, and routing logic.

The processor(s) 1110 may further include a safety cluster engine that includes a dedicated processor subsystem to handle safety management for automotive applications. The safety cluster engine may include two or more processor cores, a tightly coupled RAM, support peripherals (e.g., timers, an interrupt controller, etc.), and/or routing logic. In a safety mode, the two or more cores may operate in a lockstep mode and function as a single core with comparison logic to detect any differences between their operations.

The processor(s) 1110 may further include a real-time camera engine that may include a dedicated processor subsystem for handling real-time camera management.

The processor(s) 1110 may further include a high-dynamic range signal processor that may include an image signal processor that is a hardware engine that is part of the camera processing pipeline.

The processor(s) 1110 may include a video image compositor that may be a processing block (e.g., implemented on a microprocessor) that implements video post-processing functions needed by a video playback application to produce the final image for the player window. The video image compositor may perform lens distortion correction on wide-view camera(s) 1170, surround camera(s) 1174, and/or on in-cabin monitoring camera sensors. In-cabin monitoring camera sensor is preferably monitored by a neural network running on another instance of the Advanced SoC, configured to identify in cabin events and respond accordingly. An in-cabin system may perform lip reading to activate cellular service and place a phone call, dictate emails, change the vehicle's destination, activate or change the vehicle's infotainment system and settings, or provide voice-activated web surfing. Certain functions are available to the driver only when the vehicle is operating in an autonomous mode, and are disabled otherwise.

The video image compositor may include enhanced temporal noise reduction for both spatial and temporal noise

reduction. For example, where motion occurs in a video, the noise reduction weights spatial information appropriately, decreasing the weight of information provided by adjacent frames. Where an image or portion of an image does not include motion, the temporal noise reduction performed by the video image compositor may use information from the previous image to reduce noise in the current image.

The video image compositor may also be configured to perform stereo rectification on input stereo lens frames. The video image compositor may further be used for user interface composition when the operating system desktop is in use, and the GPU(s) **1108** is not required to continuously render new surfaces. Even when the GPU(s) **1108** is powered on and active doing 3D rendering, the video image compositor may be used to offload the GPU(s) **1108** to improve performance and responsiveness.

The SoC(s) **1104** may further include a mobile industry processor interface (MIPI) camera serial interface for receiving video and input from cameras, a high-speed interface, and/or a video input block that may be used for camera and related pixel input functions. The SoC(s) **1104** may further include an input/output controller(s) that may be controlled by software and may be used for receiving I/O signals that are uncommitted to a specific role.

The SoC(s) **1104** may further include a broad range of peripheral interfaces to enable communication with peripherals, audio codecs, power management, and/or other devices. The SoC(s) **1104** may be used to process data from cameras (e.g., connected over Gigabit Multimedia Serial Link and Ethernet), sensors (e.g., LIDAR sensor(s) **1164**, RADAR sensor(s) **1160**, etc. that may be connected over Ethernet), data from bus **1102** (e.g., speed of vehicle **102**, steering wheel position, etc.), data from GNSS sensor(s) **1158** (e.g., connected over Ethernet or CAN bus). The SoC(s) **1104** may further include dedicated high-performance mass storage controllers that may include their own DMA engines, and that may be used to free the CPU(s) **1106** from routine data management tasks.

The SoC(s) **1104** may be an end-to-end platform with a flexible architecture that spans automation levels 3-5, thereby providing a comprehensive functional safety architecture that leverages and makes efficient use of computer vision and ADAS techniques for diversity and redundancy, provides a platform for a flexible, reliable driving software stack, along with deep learning tools. The SoC(s) **1104** may be faster, more reliable, and even more energy-efficient and space-efficient than conventional systems. For example, the accelerator(s) **1114**, when combined with the CPU(s) **1106**, the GPU(s) **1108**, and the data store(s) **1116**, may provide for a fast, efficient platform for level 3-5 autonomous vehicles.

The technology thus provides capabilities and functionality that cannot be achieved by conventional systems. For example, computer vision algorithms may be executed on CPUs, which may be configured using high-level programming language, such as the C programming language, to execute a wide variety of processing algorithms across a wide variety of visual data. However, CPUs are oftentimes unable to meet the performance requirements of many computer vision applications, such as those related to execution time and power consumption, for example. In particular, many CPUs are unable to execute complex object detection algorithms in real-time, which is a requirement of in-vehicle ADAS applications, and a requirement for practical Level 3-5 autonomous vehicles.

In contrast to conventional systems, by providing a CPU complex, GPU complex, and a hardware acceleration cluster, the technology described herein allows for multiple neural networks to be performed simultaneously and/or sequentially, and for the results to be combined together to enable Level 3-5 autonomous driving functionality. For example, a CNN executing on the DLA or dGPU (e.g., the GPU(s) **1120**) may include a text and word recognition, allowing the supercomputer to read and understand traffic signs, including signs for which the neural network has not been specifically trained. The DLA may further include a neural network that is able to identify, interpret, and provides semantic understanding of the sign, and to pass that semantic understanding to the path planning modules running on the CPU Complex.

As another example, multiple neural networks may be run simultaneously, as is required for Level 3, 4, or 5 driving. For example, a warning sign consisting of "Caution: flashing lights indicate icy conditions," along with an electric light, may be independently or collectively interpreted by several neural networks. The sign itself may be identified as a traffic sign by a first deployed neural network (e.g., a neural network that has been trained), the text "Flashing lights indicate icy conditions" may be interpreted by a second deployed neural network, which informs the vehicle's path planning software (preferably executing on the CPU Complex) that when flashing lights are detected, icy conditions exist. The flashing light may be identified by operating a third deployed neural network over multiple frames, informing the vehicle's path-planning software of the presence (or absence) of flashing lights. All three neural networks may run simultaneously, such as within the DLA and/or on the GPU(s) **1108**.

In some examples, a CNN for facial recognition and vehicle owner identification may use data from camera sensors to identify the presence of an authorized driver and/or owner of the vehicle **102**. The always on sensor processing engine may be used to unlock the vehicle when the owner approaches the driver door and turn on the lights, and, in security mode, to disable the vehicle when the owner leaves the vehicle. In this way, the SoC(s) **1104** provide for security against theft and/or carjacking.

In another example, a CNN for emergency vehicle detection and identification may use data from microphones **1196** to detect and identify emergency vehicle sirens. In contrast to conventional systems, that use general classifiers to detect sirens and manually extract features, the SoC(s) **1104** use the CNN for classifying environmental and urban sounds, as well as classifying visual data. In a preferred embodiment, the CNN running on the DLA is trained to identify the relative closing speed of the emergency vehicle (e.g., by using the Doppler effect). The CNN may also be trained to identify emergency vehicles specific to the local area in which the vehicle is operating, as identified by GNSS sensor(s) **1158**. Thus, for example, when operating in Europe the CNN will seek to detect European sirens, and when in the United States the CNN will seek to identify only North American sirens. Once an emergency vehicle is detected, a control program may be used to execute an emergency vehicle safety routine, slowing the vehicle, pulling over to the side of the road, parking the vehicle, and/or idling the vehicle, with the assistance of ultrasonic sensors **1162**, until the emergency vehicle(s) passes.

The vehicle may include a CPU(s) **1118** (e.g., discrete CPU(s), or dCPU(s)), that may be coupled to the SoC(s) **1104** via a high-speed interconnect (e.g., PCIe). The CPU(s) **1118** may include an X86 processor, for example. The CPU(s) **1118** may be used to perform any of a variety of functions, including arbitrating potentially inconsistent results between ADAS sensors and the SoC(s) **1104**, and/or

monitoring the status and health of the controller(s) **1136** and/or infotainment SoC **1130**, for example.

The vehicle **102** may include a GPU(s) **1120** (e.g., discrete GPU(s), or dGPU(s)), that may be coupled to the SoC(s) **1104** via a high-speed interconnect (e.g., NVIDIA's NVLINK). The GPU(s) **1120** may provide additional artificial intelligence functionality, such as by executing redundant and/or different neural networks, and may be used to train and/or update neural networks based on input (e.g., sensor data) from sensors of the vehicle **102**.

The vehicle **102** may further include the network interface **1124** which may include one or more wireless antennas **1126** (e.g., one or more wireless antennas for different communication protocols, such as a cellular antenna, a Bluetooth antenna, etc.). The network interface **1124** may be used to enable wireless connectivity over the Internet with the cloud (e.g., with the server(s) **1178** and/or other network devices), with other vehicles, and/or with computing devices (e.g., client devices of passengers). To communicate with other vehicles, a direct link may be established between the two vehicles and/or an indirect link may be established (e.g., across networks and over the Internet). Direct links may be provided using a vehicle-to-vehicle communication link. The vehicle-to-vehicle communication link may provide the vehicle **102** information about vehicles in proximity to the vehicle **102** (e.g., vehicles in front of, on the side of, and/or behind the vehicle **102**). This functionality may be part of a cooperative adaptive cruise control functionality of the vehicle **102**.

The network interface **1124** may include a SoC that provides modulation and demodulation functionality and enables the controller(s) **1136** to communicate over wireless networks. The network interface **1124** may include a radio frequency front-end for up-conversion from baseband to radio frequency, and down conversion from radio frequency to baseband. The frequency conversions may be performed through well-known processes, and/or may be performed using super-heterodyne processes. In some examples, the radio frequency front end functionality may be provided by a separate chip. The network interface may include wireless functionality for communicating over LTE, WCDMA, UMTS, GSM, CDMA2000, Bluetooth, Bluetooth LE, Wi-Fi, Z-Wave, ZigBee, LoRaWAN, and/or other wireless protocols.

The vehicle **102** may further include data store(s) **1128** which may include off-chip (e.g., off the SoC(s) **1104**) storage. The data store(s) **1128** may include one or more storage elements including RAM, SRAM, DRAM, VRAM, Flash, hard disks, and/or other components and/or devices that may store at least one bit of data.

The vehicle **102** may further include GNSS sensor(s) **1158**. The GNSS sensor(s) **1158** (e.g., GPS and/or assisted GPS sensors), to assist in mapping, perception, occupancy grid generation, and/or path planning functions. Any number of GNSS sensor(s) **1158** may be used, including, for example and without limitation, a GPS using a USB connector with an Ethernet to Serial (RS-232) bridge.

The vehicle **102** may further include RADAR sensor(s) **1160**. The RADAR sensor(s) **1160** may be used by the vehicle **102** for long-range vehicle detection, even in darkness and/or severe weather conditions. RADAR functional safety levels may be ASIL B. The RADAR sensor(s) **1160** may use the CAN and/or the bus **1102** (e.g., to transmit data generated by the RADAR sensor(s) **1160**) for control and to access object tracking data, with access to Ethernet to access raw data in some examples. A wide variety of RADAR sensor types may be used. For example, and without limi-

tation, the RADAR sensor(s) **1160** may be suitable for front, rear, and side RADAR use. In some example, Pulse Doppler RADAR sensor(s) are used.

The RADAR sensor(s) **1160** may include different configurations, such as long range with narrow field of view, short range with wide field of view, short range side coverage, etc. In some examples, long-range RADAR may be used for adaptive cruise control functionality. The long-range RADAR systems may provide a broad field of view realized by two or more independent scans, such as within a 250 m range. The RADAR sensor(s) **1160** may help in distinguishing between static and moving objects, and may be used by ADAS systems for emergency brake assist and forward collision warning. Long-range RADAR sensors may include monostatic multimodal RADAR with multiple (e.g., six or more) fixed RADAR antennae and a high-speed CAN and FlexRay interface. In an example with six antennae, the central four antennae may create a focused beam pattern, designed to record the vehicle's **102** surroundings at higher speeds with minimal interference from traffic in adjacent lanes. The other two antennae may expand the field of view, making it possible to quickly detect vehicles entering or leaving the vehicle's **102** lane.

Mid-range RADAR systems may include, as an example, a range of up to 1160 m (front) or 80 m (rear), and a field of view of up to 42 degrees (front) or 1150 degrees (rear). Short-range RADAR systems may include, without limitation, RADAR sensors designed to be installed at both ends of the rear bumper. When installed at both ends of the rear bumper, such a RADAR sensor systems may create two beams that constantly monitor the blind spot in the rear and next to the vehicle.

Short-range RADAR systems may be used in an ADAS system for blind spot detection and/or lane change assist.

The vehicle **102** may further include ultrasonic sensor(s) **1162**. The ultrasonic sensor(s) **1162**, which may be positioned at the front, back, and/or the sides of the vehicle **102**, may be used for park assist and/or to create and update an occupancy grid. A wide variety of ultrasonic sensor(s) **1162** may be used, and different ultrasonic sensor(s) **1162** may be used for different ranges of detection (e.g., 2.5 m, 4 m). The ultrasonic sensor(s) **1162** may operate at functional safety levels of ASIL B.

The vehicle **102** may include LIDAR sensor(s) **1164**. The LIDAR sensor(s) **1164** may be used for object and pedestrian detection, emergency braking, collision avoidance, and/or other functions. The LIDAR sensor(s) **1164** may be functional safety level ASIL B. In some examples, the vehicle **102** may include multiple LIDAR sensors **1164** (e.g., two, four, six, etc.) that may use Ethernet (e.g., to provide data to a Gigabit Ethernet switch).

In some examples, the LIDAR sensor(s) **1164** may be capable of providing a list of objects and their distances for a 360-degree field of view. Commercially available LIDAR sensor(s) **1164** may have an advertised range of approximately 102 m, with an accuracy of 2 cm-3 cm, and with support for a 102 Mbps Ethernet connection, for example. In some examples, one or more non-protruding LIDAR sensors **1164** may be used. In such examples, the LIDAR sensor(s) **1164** may be implemented as a small device that may be embedded into the front, rear, sides, and/or corners of the vehicle **102**. The LIDAR sensor(s) **1164**, in such examples, may provide up to a 1120-degree horizontal and 35-degree vertical field-of-view, with a 200 m range even for low-reflectivity objects. Front-mounted LIDAR sensor(s) **1164** may be configured for a horizontal field of view between 45 degrees and 135 degrees.

In some examples, LIDAR technologies, such as 3D flash LIDAR, may also be used. 3D Flash LIDAR uses a flash of a laser as a transmission source, to illuminate vehicle surroundings up to approximately 200 m. A flash LIDAR unit includes a receptor, which records the laser pulse transit time and the reflected light on each pixel, which in turn corresponds to the range from the vehicle to the objects. Flash LIDAR may allow for highly accurate and distortion-free images of the surroundings to be generated with every laser flash. In some examples, four flash LIDAR sensors may be deployed, one at each side of the vehicle 102. Available 3D flash LIDAR systems include a solid-state 3D staring array LIDAR camera with no moving parts other than a fan (e.g., a non-scanning LIDAR device). The flash LIDAR device may use a 5 nanosecond class I (eye-safe) laser pulse per frame and may capture the reflected laser light in the form of 3D range point clouds and co-registered intensity data. By using flash LIDAR, and because flash LIDAR is a solid-state device with no moving parts, the LIDAR sensor(s) 1164 may be less susceptible to motion blur, vibration, and/or shock.

The vehicle may further include IMU sensor(s) 1166. The IMU sensor(s) 1166 may be located at a center of the rear axle of the vehicle 102, in some examples. The IMU sensor(s) 1166 may include, for example and without limitation, an accelerometer(s), a magnetometer(s), a gyroscope(s), a magnetic compass(es), and/or other sensor types. In some examples, such as in six-axis applications, the IMU sensor(s) 1166 may include accelerometers and gyroscopes, while in nine-axis applications, the IMU sensor(s) 1166 may include accelerometers, gyroscopes, and magnetometers.

In some embodiments, the IMU sensor(s) 1166 may be implemented as a miniature, high performance GPS-Aided Inertial Navigation System (GPS/INS) that combines micro-electro-mechanical systems (MEMS) inertial sensors, a high-sensitivity GPS receiver, and advanced Kalman filtering algorithms to provide estimates of position, velocity, and attitude. As such, in some examples, the IMU sensor(s) 1166 may enable the vehicle 102 to estimate heading without requiring input from a magnetic sensor by directly observing and correlating the changes in velocity from GPS to the IMU sensor(s) 1166. In some examples, the IMU sensor(s) 1166 and the GNSS sensor(s) 1158 may be combined in a single integrated unit.

The vehicle may include microphone(s) 1196 placed in and/or around the vehicle 102. The microphone(s) 1196 may be used for emergency vehicle detection and identification, among other things.

The vehicle may further include any number of camera types, including stereo camera(s) 1168, wide-view camera(s) 1170, infrared camera(s) 1172, surround camera(s) 1174, long-range and/or mid-range camera(s) 1198, and/or other camera types. The cameras may be used to capture image data around an entire periphery of the vehicle 102. The types of cameras used depends on the embodiments and requirements for the vehicle 102, and any combination of camera types may be used to provide the necessary coverage around the vehicle 102. In addition, the number of cameras may differ depending on the embodiment. For example, the vehicle may include six cameras, seven cameras, ten cameras, twelve cameras, and/or another number of cameras. The cameras may support, as an example and without limitation, Gigabit Multimedia Serial Link (GMSL) and/or Gigabit Ethernet. Each of the camera(s) is described with more detail herein with respect to FIG. 11A and FIG. 11B.

The vehicle 102 may further include vibration sensor(s) 1142. The vibration sensor(s) 1142 may measure vibrations of components of the vehicle, such as the axle(s). For example, changes in vibrations may indicate a change in road surfaces. In another example, when two or more vibration sensors 1142 are used, the differences between the vibrations may be used to determine friction or slippage of the road surface (e.g., when the difference in vibration is between a power-driven axle and a freely rotating axle).

The vehicle 102 may include an ADAS system 1138. The ADAS system 1138 may include a SoC, in some examples. The ADAS system 1138 may include autonomous/adaptive/automatic cruise control (ACC), cooperative adaptive cruise control (CACC), forward crash warning (FCW), automatic emergency braking (AEB), lane departure warnings (LDW), lane keep assist (LKA), blind spot warning (BSW), rear cross-traffic warning (RCTW), collision warning systems (CWS), lane centering (LC), and/or other features and functionality.

The ACC systems may use RADAR sensor(s) 1160, LIDAR sensor(s) 1164, and/or a camera(s). The ACC systems may include longitudinal ACC and/or lateral ACC. Longitudinal ACC monitors and controls the distance to the vehicle immediately ahead of the vehicle 102 and automatically adjust the vehicle speed to maintain a safe distance from vehicles ahead. Lateral ACC performs distance keeping, and advises the vehicle 102 to change lanes when necessary. Lateral ACC is related to other ADAS applications such as LCA and CWS.

CACC uses information from other vehicles that may be received via the network interface 1124 and/or the wireless antenna(s) 1126 from other vehicles via a wireless link, or indirectly, over a network connection (e.g., over the Internet). Direct links may be provided by a vehicle-to-vehicle (V2V) communication link, while indirect links may be infrastructure-to-vehicle (I2V) communication link. In general, the V2V communication concept provides information about the immediately preceding vehicles (e.g., vehicles immediately ahead of and in the same lane as the vehicle 102), while the I2V communication concept provides information about traffic further ahead. CACC systems may include either or both I2V and V2V information sources. Given the information of the vehicles ahead of the vehicle 102, CACC may be more reliable and it has potential to improve traffic flow smoothness and reduce congestion on the road.

FCW systems are designed to alert the driver to a hazard, so that the driver may take corrective action. FCW systems use a front-facing camera and/or RADAR sensor(s) 1160, coupled to a dedicated processor, DSP, FPGA, and/or ASIC, that is electrically coupled to driver feedback, such as a display, speaker, and/or vibrating component. FCW systems may provide a warning, such as in the form of a sound, visual warning, vibration and/or a quick brake pulse.

AEB systems detect an impending forward collision with another vehicle or other object, and may automatically apply the brakes if the driver does not take corrective action within a specified time or distance parameter. AEB systems may use front-facing camera(s) and/or RADAR sensor(s) 1160, coupled to a dedicated processor, DSP, FPGA, and/or ASIC. When the AEB system detects a hazard, it typically first alerts the driver to take corrective action to avoid the collision and, if the driver does not take corrective action, the AEB system may automatically apply the brakes in an effort to prevent, or at least mitigate, the impact of the predicted collision. AEB systems, may include techniques such as dynamic brake support and/or crash imminent braking.

LDW systems provide visual, audible, and/or tactile warnings, such as steering wheel or seat vibrations, to alert the driver when the vehicle **102** crosses lane markings. A LDW system does not activate when the driver indicates an intentional lane departure, by activating a turn signal. LDW systems may use front-side facing cameras, coupled to a dedicated processor, DSP, FPGA, and/or ASIC, that is electrically coupled to driver feedback, such as a display, speaker, and/or vibrating component.

LKA systems are a variation of LDW systems. LKA systems provide steering input or braking to correct the vehicle **102** if the vehicle **102** starts to exit the lane.

BSW systems detects and warn the driver of vehicles in an automobile's blind spot. BSW systems may provide a visual, audible, and/or tactile alert to indicate that merging or changing lanes is unsafe. The system may provide an additional warning when the driver uses a turn signal. BSW systems may use rear-side facing camera(s) and/or RADAR sensor(s) **1160**, coupled to a dedicated processor, DSP, FPGA, and/or ASIC, that is electrically coupled to driver feedback, such as a display, speaker, and/or vibrating component.

RCTW systems may provide visual, audible, and/or tactile notification when an object is detected outside the rear-camera range when the vehicle **102** is backing up. Some RCTW systems include AEB to ensure that the vehicle brakes are applied to avoid a crash. RCTW systems may use one or more rear-facing RADAR sensor(s) **1160**, coupled to a dedicated processor, DSP, FPGA, and/or ASIC, that is electrically coupled to driver feedback, such as a display, speaker, and/or vibrating component.

Conventional ADAS systems may be prone to false positive results which may be annoying and distracting to a driver, but typically are not catastrophic, because the ADAS systems alert the driver and allow the driver to decide whether a safety condition truly exists and act accordingly. However, in an autonomous vehicle **102**, the vehicle **102** itself must, in the case of conflicting results, decide whether to heed the result from a primary computer or a secondary computer (e.g., a first controller **1136** or a second controller **1136**). For example, in some embodiments, the ADAS system **1138** may be a backup and/or secondary computer for providing perception information to a backup computer rationality module. The backup computer rationality monitor may run a redundant diverse software on hardware components to detect faults in perception and dynamic driving tasks. Outputs from the ADAS system **1138** may be provided to a supervisory MCU. If outputs from the primary computer and the secondary computer conflict, the supervisory MCU must determine how to reconcile the conflict to ensure safe operation.

In some examples, the primary computer may be configured to provide the supervisory MCU with a confidence score, indicating the primary computer's confidence in the chosen result. If the confidence score exceeds a threshold, the supervisory MCU may follow the primary computer's direction, regardless of whether the secondary computer provides a conflicting or inconsistent result. Where the confidence score does not meet the threshold, and where the primary and secondary computer indicate different results (e.g., the conflict), the supervisory MCU may arbitrate between the computers to determine the appropriate outcome.

The supervisory MCU may be configured to run a neural network(s) that is trained and configured to determine, based on outputs from the primary computer and the secondary computer, conditions under which the secondary computer provides false alarms. Thus, the neural network(s) in the supervisory MCU may learn when the secondary computer's output may be trusted, and when it cannot. For example, when the secondary computer is a RADAR-based FCW system, a neural network(s) in the supervisory MCU may learn when the FCW system is identifying metallic objects that are not, in fact, hazards, such as a drainage grate or manhole cover that triggers an alarm. Similarly, when the secondary computer is a camera-based LDW system, a neural network in the supervisory MCU may learn to override the LDW when bicyclists or pedestrians are present and a lane departure is, in fact, the safest maneuver. In embodiments that include a neural network(s) running on the supervisory MCU, the supervisory MCU may include at least one of a DLA or GPU suitable for running the neural network(s) with associated memory. In preferred embodiments, the supervisory MCU may comprise and/or be included as a component of the SoC(s) **1104**.

In other examples, ADAS system **1138** may include a secondary computer that performs ADAS functionality using traditional rules of computer vision. As such, the secondary computer may use classic computer vision rules (if-then), and the presence of a neural network(s) in the supervisory MCU may improve reliability, safety and performance. For example, the diverse implementation and intentional non-identity makes the overall system more fault-tolerant, especially to faults caused by software (or software-hardware interface) functionality. For example, if there is a software bug or error in the software running on the primary computer, and the non-identical software code running on the secondary computer provides the same overall result, the supervisory MCU may have greater confidence that the overall result is correct, and the bug in software or hardware on primary computer is not causing material error.

In some examples, the output of the ADAS system **1138** may be fed into the primary computer's perception block and/or the primary computer's dynamic driving task block. For example, if the ADAS system **1138** indicates a forward crash warning due to an object immediately ahead, the perception block may use this information when identifying objects. In other examples, the secondary computer may have its own neural network which is trained and thus reduces the risk of false positives, as described herein.

The vehicle **102** may further include the infotainment SoC **1130** (e.g., an in-vehicle infotainment system (IVI)). Although illustrated and described as a SoC, the infotainment system may not be a SoC, and may include two or more discrete components. The infotainment SoC **1130** may include a combination of hardware and software that may be used to provide audio (e.g., music, a personal digital assistant, navigational instructions, news, radio, etc.), video (e.g., TV, movies, streaming, etc.), phone (e.g., hands-free calling), network connectivity (e.g., LTE, Wi-Fi, etc.), and/or information services (e.g., navigation systems, rear-parking assistance, a radio data system, vehicle related information such as fuel level, total distance covered, brake fuel level, oil level, door open/close, air filter information, etc.) to the vehicle **102**. For example, the infotainment SoC **1130** may radios, disk players, navigation systems, video players, USB and Bluetooth connectivity, carputers, in-car entertainment, Wi-Fi, steering wheel audio controls, hands free voice control, a heads-up display (HUD), an HMI display **1134**, a telematics device, a control panel (e.g., for controlling and/or interacting with various components, features, and/or systems), and/or other components. The infotainment SoC **1130** may further be used to provide information (e.g., visual

and/or audible) to a user(s) of the vehicle, such as information from the ADAS system **1138**, autonomous driving information such as planned vehicle maneuvers, trajectories, surrounding environment information (e.g., intersection information, vehicle information, road information, etc.), and/or other information.

The infotainment SoC **1130** may include GPU functionality. The infotainment SoC **1130** may communicate over the bus **1102** (e.g., CAN bus, Ethernet, etc.) with other devices, systems, and/or components of the vehicle **102**. In some examples, the infotainment SoC **1130** may be coupled to a supervisory MCU such that the GPU of the infotainment system may perform some self-driving functions in the event that the primary controller(s) **1136** (e.g., the primary and/or backup computers of the vehicle **102**) fail. In such an example, the infotainment SoC **1130** may put the vehicle **102** into a chauffeur to safe stop mode, as described herein.

The vehicle **102** may further include an instrument cluster **1132** (e.g., a digital dash, an electronic instrument cluster, a digital instrument panel, etc.). The instrument cluster **1132** may include a controller and/or supercomputer (e.g., a discrete controller or supercomputer). The instrument cluster **1132** may include a set of instrumentation such as a speedometer, fuel level, oil pressure, tachometer, odometer, turn indicators, gearshift position indicator, seat belt warning light(s), parking-brake warning light(s), engine-malfunction light(s), airbag (SRS) system information, lighting controls, safety system controls, navigation information, etc. In some examples, information may be displayed and/or shared among the infotainment SoC **1130** and the instrument cluster **1132**. In other words, the instrument cluster **1132** may be included as part of the infotainment SoC **1130**, or vice versa.

FIG. **11D** is a system diagram for communication between cloud-based server(s) and the example autonomous vehicle **102** of FIG. **11A**, in accordance with some embodiments of the present disclosure. The system **1176** may include server(s) **1178**, network(s) **1190**, and vehicles, including the vehicle **102**. The server(s) **1178** may include a plurality of GPUs **1184(A)-1184(H)** (collectively referred to herein as GPUs **1184**), PCIe switches **1182(A)-1182(H)** (collectively referred to herein as PCIe switches **1182**), and/or CPUs **1180(A)-1180(B)** (collectively referred to herein as CPUs **1180**). The GPUs **1184**, the CPUs **1180**, and the PCIe switches may be interconnected with high-speed interconnects such as, for example and without limitation, NVLink interfaces **1188** developed by NVIDIA and/or PCIe connections **1186**. In some examples, the GPUs **1184** are connected via NVLink and/or NVSwitch SoC and the GPUs **1184** and the PCIe switches **1182** are connected via PCIe interconnects. Although eight GPUs **1184**, two CPUs **1180**, and two PCIe switches are illustrated, this is not intended to be limiting. Depending on the embodiment, each of the server(s) **1178** may include any number of GPUs **1184**, CPUs **1180**, and/or PCIe switches. For example, the server(s) **1178** may each include eight, sixteen, thirty-two, and/or more GPUs **1184**.

The server(s) **1178** may receive, over the network(s) **1190** and from the vehicles, image data representative of images showing unexpected or changed road conditions, such as recently commenced road-work. The server(s) **1178** may transmit, over the network(s) **1190** and to the vehicles, neural networks **1192**, updated neural networks **1192**, and/or map information **1194**, including information regarding traffic and road conditions. The updates to the map information **1194** may include updates for the HD map **1122**, such as information regarding construction sites, potholes, detours,

flooding, and/or other obstructions. In some examples, the neural networks **1192**, the updated neural networks **1192**, and/or the map information **1194** may have resulted from new training and/or experiences represented in data received from any number of vehicles in the environment, and/or based on training performed at a datacenter (e.g., using the server(s) **1178** and/or other servers).

The server(s) **1178** may be used to train machine learning models (e.g., neural networks) based on training data. The training data may be generated by the vehicles, and/or may be generated in a simulation (e.g., using a game engine). In some examples, the training data is tagged (e.g., where the neural network benefits from supervised learning) and/or undergoes other pre-processing, while in other examples the training data is not tagged and/or pre-processed (e.g., where the neural network does not require supervised learning). Once the machine learning models are trained, the machine learning models may be used by the vehicles (e.g., transmitted to the vehicles over the network(s) **1190**, and/or the machine learning models may be used by the server(s) **1178** to remotely monitor the vehicles.

In some examples, the server(s) **1178** may receive data from the vehicles and apply the data to up-to-date real-time neural networks for real-time intelligent inferencing. The server(s) **1178** may include deep-learning supercomputers and/or dedicated AI computers powered by GPU(s) **1184**, such as a DGX and DGX Station machines developed by NVIDIA. However, in some examples, the server(s) **1178** may include deep learning infrastructure that use only CPU-powered datacenters.

The deep-learning infrastructure of the server(s) **1178** may be capable of fast, real-time inferencing, and may use that capability to evaluate and verify the health of the processors, software, and/or associated hardware in the vehicle **102**. For example, the deep-learning infrastructure may receive periodic updates from the vehicle **102**, such as a sequence of images and/or objects that the vehicle **102** has located in that sequence of images (e.g., via computer vision and/or other machine learning object classification techniques). The deep-learning infrastructure may run its own neural network to identify the objects and compare them with the objects identified by the vehicle **102** and, if the results do not match and the infrastructure concludes that the AI in the vehicle **102** is malfunctioning, the server(s) **1178** may transmit a signal to the vehicle **102** instructing a fail-safe computer of the vehicle **102** to assume control, notify the passengers, and complete a safe parking maneuver.

For inferencing, the server(s) **1178** may include the GPU(s) **1184** and one or more programmable inference accelerators (e.g., NVIDIA's TensorRT 3). The combination of GPU-powered servers and inference acceleration may make real-time responsiveness possible. In other examples, such as where performance is less critical, servers powered by CPUs, FPGAs, and other processors may be used for inferencing.

Example Computing Device

FIG. **12** is a block diagram of an example computing device **1200** suitable for use in implementing some embodiments of the present disclosure. Computing device **1200** may include a bus **1202** that directly or indirectly couples the following devices: memory **1204**, one or more central processing units (CPUs) **1206**, one or more graphics processing units (GPUs) **1208**, a communication interface **1210**, input/output (I/O) ports **1212**, input/output components **1214**, a power supply **1216**, and one or more presentation components **1218** (e.g., display(s)).

Although the various blocks of FIG. **12** are shown as connected via the bus **1202** with lines, this is not intended to be limiting and is for clarity only. For example, in some embodiments, a presentation component **1218**, such as a display device, may be considered an I/O component **1214** (e.g., if the display is a touch screen). As another example, the CPUs **1206** and/or GPUs **1208** may include memory (e.g., the memory **1204** may be representative of a storage device in addition to the memory of the GPUs **1208**, the CPUs **1206**, and/or other components). In other words, the computing device of FIG. **12** is merely illustrative. Distinction is not made between such categories as "workstation," "server," "laptop," "desktop," "tablet," "client device," "mobile device," "hand-held device," "game console," "electronic control unit (ECU)," "virtual reality system," and/or other device or system types, as all are contemplated within the scope of the computing device of FIG. **12**.

The bus **1202** may represent one or more busses, such as an address bus, a data bus, a control bus, or a combination thereof. The bus **1202** may include one or more bus types, such as an industry standard architecture (ISA) bus, an extended industry standard architecture (EISA) bus, a video electronics standards association (VESA) bus, a peripheral component interconnect (PCI) bus, a peripheral component interconnect express (PCIe) bus, and/or another type of bus.

The memory **1204** may include any of a variety of computer-readable media. The computer-readable media may be any available media that may be accessed by the computing device **1200**. The computer-readable media may include both volatile and nonvolatile media, and removable and non-removable media. By way of example, and not limitation, the computer-readable media may comprise computer-storage media and communication media.

The computer-storage media may include both volatile and nonvolatile media and/or removable and non-removable media implemented in any method or technology for storage of information such as computer-readable instructions, data structures, program modules, and/or other data types. For example, the memory **1204** may store computer-readable instructions (e.g., that represent a program(s) and/or a program element(s), such as an operating system. Computer-storage media may include, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical disk storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which may be used to store the desired information and which may be accessed by computing device **1200**. As used herein, computer storage media does not comprise signals per se.

The communication media may embody computer-readable instructions, data structures, program modules, and/or other data types in a modulated data signal such as a carrier wave or other transport mechanism and includes any information delivery media. The term "modulated data signal" may refer to a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, the communication media may include wired media such as a wired network or direct-wired connection, and wireless media such as acoustic, RF, infrared and other wireless media. Combinations of any of the above should also be included within the scope of computer-readable media.

The CPU(s) **1206** may be configured to execute the computer-readable instructions to control one or more components of the computing device **1200** to perform one or more of the methods and/or processes described herein. The

CPU(s) **1206** may each include one or more cores (e.g., one, two, four, eight, twenty-eight, seventy-two, etc.) that are capable of handling a multitude of software threads simultaneously. The CPU(s) **1206** may include any type of processor, and may include different types of processors depending on the type of computing device **1200** implemented (e.g., processors with fewer cores for mobile devices and processors with more cores for servers). For example, depending on the type of computing device **1200**, the processor may be an ARM processor implemented using Reduced Instruction Set Computing (RISC) or an x86 processor implemented using Complex Instruction Set Computing (CISC). The computing device **1200** may include one or more CPUs **1206** in addition to one or more microprocessors or supplementary co-processors, such as math co-processors.

The GPU(s) **1208** may be used by the computing device **1200** to render graphics (e.g., 3D graphics). The GPU(s) **1208** may include hundreds or thousands of cores that are capable of handling hundreds or thousands of software threads simultaneously. The GPU(s) **1208** may generate pixel data for output images in response to rendering commands (e.g., rendering commands from the CPU(s) **1206** received via a host interface). The GPU(s) **1208** may include graphics memory, such as display memory, for storing pixel data. The display memory may be included as part of the memory **1204**. The GPU(s) **708** may include two or more GPUs operating in parallel (e.g., via a link). When combined together, each GPU **1208** may generate pixel data for different portions of an output image or for different output images (e.g., a first GPU for a first image and a second GPU for a second image). Each GPU may include its own memory, or may share memory with other GPUs.

In examples where the computing device **1200** does not include the GPU(s) **1208**, the CPU(s) **1206** may be used to render graphics.

The communication interface **1210** may include one or more receivers, transmitters, and/or transceivers that enable the computing device **700** to communicate with other computing devices via an electronic communication network, included wired and/or wireless communications. The communication interface **1210** may include components and functionality to enable communication over any of a number of different networks, such as wireless networks (e.g., Wi-Fi, Z-Wave, Bluetooth, Bluetooth LE, ZigBee, etc.), wired networks (e.g., communicating over Ethernet), low-power wide-area networks (e.g., LoRaWAN, SigFox, etc.), and/or the Internet.

The I/O ports **1212** may enable the computing device **1200** to be logically coupled to other devices including the I/O components **1214**, the presentation component(s) **1218**, and/or other components, some of which may be built in to (e.g., integrated in) the computing device **1200**. Illustrative I/O components **1214** include a microphone, mouse, keyboard, joystick, game pad, game controller, satellite dish, scanner, printer, wireless device, etc. The I/O components **1214** may provide a natural user interface (NUI) that processes air gestures, voice, or other physiological inputs generated by a user. In some instances, inputs may be transmitted to an appropriate network element for further processing. An NUI may implement any combination of speech recognition, stylus recognition, facial recognition, biometric recognition, gesture recognition both on screen and adjacent to the screen, air gestures, head and eye tracking, and touch recognition (as described in more detail below) associated with a display of the computing device **1200**. The computing device **1200** may be include depth

cameras, such as stereoscopic camera systems, infrared camera systems, RGB camera systems, touchscreen technology, and combinations of these, for gesture detection and recognition. Additionally, the computing device **1200** may include accelerometers or gyroscopes (e.g., as part of an inertia measurement unit (IMU)) that enable detection of motion. In some examples, the output of the accelerometers or gyroscopes may be used by the computing device **1200** to render immersive augmented reality or virtual reality.

The power supply **1216** may include a hard-wired power supply, a battery power supply, or a combination thereof. The power supply **1216** may provide power to the computing device **1200** to enable the components of the computing device **1200** to operate.

The presentation component(s) **1218** may include a display (e.g., a monitor, a touch screen, a television screen, a heads-up-display (HUD), other display types, or a combination thereof), speakers, and/or other presentation components. The presentation component(s) **1218** may receive data from other components (e.g., the GPU(s) **1208**, the CPU(s) **1206**, etc.), and output the data (e.g., as an image, video, sound, etc.).

The disclosure may be described in the general context of computer code or machine-useable instructions, including computer-executable instructions such as program modules, being executed by a computer or other machine, such as a personal data assistant or other handheld device. Generally, program modules including routines, programs, objects, components, data structures, etc., refer to code that perform particular tasks or implement particular abstract data types. The disclosure may be practiced in a variety of system configurations, including hand-held devices, consumer electronics, general-purpose computers, more specialty computing devices, etc. The disclosure may also be practiced in distributed computing environments where tasks are performed by remote-processing devices that are linked through a communications network.

As used herein, a recitation of "and/or" with respect to two or more elements should be interpreted to mean only one element, or a combination of elements. For example, "element A, element B, and/or element C" may include only element A, only element B, only element C, element A and element B, element A and element C, element B and element C, or elements A, B, and C. In addition, "at least one of element A or element B" may include at least one of element A, at least one of element B, or at least one of element A and at least one of element B. Further, "at least one of element A and element B" may include at least one of element A, at least one of element B, or at least one of element A and at least one of element B.

The subject matter of the present disclosure is described with specificity herein to meet statutory requirements. However, the description itself is not intended to limit the scope of this disclosure. Rather, the inventors have contemplated that the claimed subject matter might also be embodied in other ways, to include different steps or combinations of steps similar to the ones described in this document, in conjunction with other present or future technologies. Moreover, although the terms "step" and/or "block" may be used herein to connote different elements of methods employed, the terms should not be interpreted as implying any particular order among or between various steps herein disclosed unless and except when the order of individual steps is explicitly described.

What is claimed is:

1. A method comprising:

receiving, using a simulator component, simulation data representing a simulated environment corresponding to a virtual representation of an autonomous or semi-autonomous machine;

generating, using the simulator component and based at least on the simulation data, virtual sensor data representing the simulated environment perceived using at least one virtual sensor of the virtual representation of the autonomous or semi-autonomous machine within the simulated environment;

receiving, using a computing system of a hardware-in-the-loop (HIL) device and from the simulator component, the virtual sensor data, the HIL device including hardware corresponding to the autonomous or semi-autonomous machine;

applying the virtual sensor data to one or more machine learning models being executed using the computing system of the HIL device;

determining, using the one or more machine learning models being executed using the computing system of the HIL device and based at least on the virtual sensor data, at least one operation for the virtual representation; and

transmitting, using the computing system and to the simulator component, operative data representing the at least one operation for the virtual representation.

2. The method of claim **1**, wherein the virtual sensor data is generated using simulation software executed using one or more computing devices.

3. The method of claim **1**, wherein:

the computing system of the HIL device that includes the hardware corresponding to the autonomous or semi-autonomous machine is included in a vehicle simulator component; and

the vehicle simulator component communicates with the simulator component to receive the simulation data and transmit the operative data.

4. The method of claim **2**, wherein:

when integrated into an autonomous or semi-autonomous machine, the computing system communicates with one or more other components of the autonomous or semi-autonomous machine using at least one of a communication type or a communication protocol; and

during a simulation corresponding to the simulation data, the computing system communicates with the one or more computing devices using the at least one of the communication type or the communication protocol.

5. The method of claim **1**, further comprising:

generating encoded sensor data by encoding the virtual sensor data using a sensor data format that corresponds to real-world sensor data generated using a real-world sensor of the autonomous or semi-autonomous machine,

wherein the determining the at least one operation is based at least on the encoded sensor data.

6. The method of claim **1**, wherein the virtual sensor data includes a sensor data format that corresponds to real-world sensor data generated using a real-world sensor of the autonomous or semi-autonomous machine.

7. The method of claim **1**, further comprising:

generating, using the simulation data, additional virtual sensor data representing the simulated environment perceived by at least one additional virtual sensor of the virtual representation; and

determining, using the one or more machine learning models executed using the computing system of the

HIL device, at least one additional operation for the virtual representation based at least on the additional virtual sensor data,

wherein the operative data further represents the at least one additional operation for the virtual representation.

8. The method of claim 1, wherein the virtual representation comprises at least one of a virtual vehicle or a virtual robot that simulates the autonomous or semi-autonomous machine.

9. The method of claim 1, wherein the simulation data includes virtual data and real-world data, and the simulated environment includes one or more representations corresponding to the virtual data and one or more representations corresponding to the real-world data.

10. A method comprising:

obtaining virtual sensor data representing a simulated environment perceived by at least one virtual sensor of a virtual representation of an autonomous or semi-autonomous machine within the simulated environment, the virtual sensor data corresponding to a first data format;

generating, based at least on the virtual sensor data corresponding to the first data format, encoded sensor data by encoding the virtual sensor data using a second data format that is associated with at least one of one or more machine learning models or a hardware-in-the-loop (HIL) device that includes hardware corresponding to the autonomous or semi-autonomous machine;

determining, using the HIL device and based at least on the one or more machine learning models, at least one operation for the virtual representation based at least on the encoded sensor data; and

transmitting operative data representing the at least one operation of the virtual representation.

11. The method of claim 10, wherein:

the HIL device is included in a vehicle simulator component; and

the vehicle simulator component communicates with a simulation component to perform the transmitting of the operative data.

12. The method of claim 10, wherein:

when in deployment within the autonomous or semi-autonomous machine, the hardware communicates with one or more other components of the autonomous or semi-autonomous machine using at least one of a communication type or a communication protocol; and

during simulation, the hardware communicates with one or more computing devices to obtain at least one of the virtual sensor data or encoded sensor data using the at least one of the communication type or the communication protocol.

13. The method of claim 10, wherein the at least one of virtual sensor data or the encoded sensor data is generated using simulation software executed using a one or more computing devices.

14. The method of claim 10, wherein the virtual representation corresponds to at least one of a virtual vehicle or a virtual robot that simulates the autonomous or semi-autonomous machine.

15. The method of claim 10, wherein the simulated environment includes representations of virtual data augmented with representations of real-world data.

16. The method of claim 10, wherein the second data format of the encoded sensor data is associated with a driving software stack that is executed using the HIL device that includes the hardware corresponding to the autonomous

or semi-autonomous machine, the driving software stack including the one or more machine learning models.

17. The method of claim 10, wherein:

the generating of the encoded sensor data by encoding the virtual sensor data is performed using a simulation component; and

the method further comprises receiving, using one or more computing devices associated with the HIL device, the virtual sensor data from the simulation component.

18. A computing device comprising one or more processing units to:

receive, from one or more computing devices, virtual sensor data representing a simulated environment perceived using at least one virtual sensor of a virtual representation of an autonomous or semi-autonomous machine within the simulated environment, the computing device being part of a hardware-in-the-loop (HIL) device that includes hardware for installation in the autonomous or semi-autonomous machine;

determine, using one or more machine learning models, at least one operation for the virtual representation based at least on the virtual sensor data; and

transmit, to at least one computing device of the one or more computing devices, operative data representing the at least one operation of the virtual representation.

19. The computing device of claim 18, wherein:

when installed in the autonomous or semi-autonomous machine, the hardware communicates with one or more other components of the autonomous or semi-autonomous machine using at least one of a communication type or a communication protocol; and

during simulation, the computing device communicates with the one or more computing devices using the at least one of the communication type or the communication protocol.

20. The computing device of claim 18, wherein the computing device is comprised in at least one of:

a control system for an autonomous or semi-autonomous machine;

a perception system for an autonomous or semi-autonomous machine;

a system for performing simulation operations;

a system for performing digital twin operations;

a system for performing real-time streaming;

a system for generating or presenting virtual reality (VR) content;

a system for generating or presenting augmented reality (AR) content;

a system for generating or presenting mixed reality (MR) content;

a system for performing light transport simulation;

a system for performing collaborative content creation for 3D assets;

a system for performing deep learning operations;

a system implemented using an edge device;

a system implemented using a robot;

a system for performing conversational AI operations;

a system for generating synthetic data;

a system incorporating one or more virtual machines (VMs);

a system implemented at least partially in a data center; or

a system implemented at least partially using cloud computing resources.

* * * * *