(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
19 June 2025 (19.06.2025)

WIPO | PCT

(10) International Publication Number
WO 2025/129093 A1

(51) International Patent Classification:
G06N 3/045 (2023.01) *G06T 15/00* (2011.01)
G06N 3/084 (2023.01)

(21) International Application Number:
PCT/US2024/060162

(22) International Filing Date:
13 December 2024 (13.12.2024)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
63/609,690 13 December 2023 (13.12.2023) US

(71) Applicant: **GOOGLE LLC** [US/US]; 1600 Amphitheatre Parkway, Mountain View, California 94043 (US).

(72) Inventors: **DUCKWORTH, Daniel Christopher**; c/o Google LLC, Tucholskystraße 2, 10117 Berlin (DE). **REISER, Christian Johannes**; c/o Google LLC, Tucholskystraße 2, 10117 Berlin (DE). **SZELISKI, Richard Stephen**; c/o Google LLC, 1600 Amphitheatre Parkway, Mountain View, California 94043 (US). **ZHIZHIN, Petr Nikolayevich**; c/o Google LLC, Tucholskystraße 2, 10117 Berlin (DE). **THIBERT, Jean-François**; c/o Google LLC, 425 Av. Viger O, Montréal, Québec QC H2Z 1W5 (CA). **BARRON, Jonathan Tilton**; c/o Google LLC, 1600 Amphitheatre Parkway, Mountain View, California 94043 (US). **LUČIĆ, Mario**; c/o Google LLC, Brandschenkestrasse 110, 8002 Zurich (CH). **HEDMAN, Lars Peter Johannes**; c/o Google LLC, 6 Pancras Square, London Greater London N1C 4AG (GB).

(74) Agent: **PROBST, Joseph J.** et al.; DORITY & MANNING, P.A., P.O. BOX 1449, GREENVILLE, South Carolina 29602 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CV, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IQ, IR, IS, IT, JM, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, MG, MK, MN, MU, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, CV, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SC, SD, SL, ST,

(54) Title: RADIANCE FIELD DISTILLATION FOR TRAINING OF REAL-TIME RADIANCE FIELD MODELS

(57) Abstract: Provided are systems and methods to train neural radiance fields for improved rendering latency. The proposed framework can include the use of a 'teacher' model and a 'student' model. The teacher model is a neural radiance field model that has been previously trained to model a three-dimensional scene. The student model, on the other hand, is a neural radiance field model that is yet to be trained. This method is designed to improve the training of neural radiance fields for real-time rendering, which is a significant challenge in the field of computer graphics.

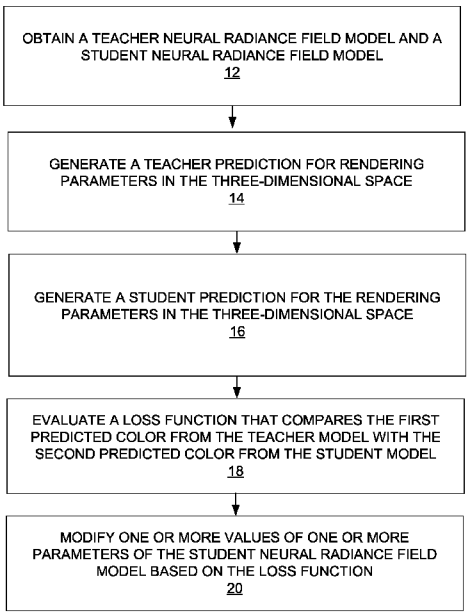


Figure 1

WO 2025/129093 A1

[Continued on next page]

SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, ME, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *of inventorship (Rule 4.17(iv))*

Published:

— *with international search report (Art. 21(3))*

RADIANCE FIELD DISTILLATION FOR TRAINING OF REAL-TIME RADIANCE FIELD MODELS

RELATED APPLICATIONS

[0001] This application claims priority to and the benefit of United States Provisional Patent Application Number 63/609,690, filed December 13, 2023. United States Provisional Patent Application Number 63/609,690 is hereby incorporated by reference in its entirety.

FIELD

[0002] The present disclosure relates generally to machine learning models known as neural radiance fields. More particularly, the present disclosure relates to a training framework for distilling a state-of-the-art neural radiance field model into a student radiance field model amenable to real-time rendering.

BACKGROUND

[0003] In the field of computer graphics and machine learning, the use of neural radiance fields (NeRF) has emerged as a powerful tool for synthesizing novel views of three-dimensional (3D) scenes from 2D images. These models are trained on a dataset of 2D images of a scene, and once trained, can be used to generate photorealistic 3D renderings of the scene from any viewpoint. However, a significant challenge with the use of NeRF models lies in their computational intensity, which often results in a latency between the input of a new viewpoint and the output of the corresponding 3D rendering. This latency can be prohibitive for real-time applications, such as virtual reality (VR) or augmented reality (AR), where instantaneous rendering is crucial for user immersion and interaction.

[0004] In order to achieve real-time frame rates, existing solutions often adopt simplified model architectures that minimize the amount of computation required at inference time. These simplified models typically either eliminate the use of neural networks or cache the outputs of the network in a regular grid. However, these simplifications result in a loss of the inductive bias provided by the architectures of more computationally intensive NeRF models, which leverage neural networks as a core part of their inference pipeline. Consequently, these simplified models often suffer from distracting artifacts such as floating or implausible geometry and miscolorations, impairing the quality of the rendered graphics.

[0005] Moreover, existing real-time NeRF models are typically trained on the same datasets as their more computationally intensive counterparts, namely sets of 2D photos

coupled with intrinsic and extrinsic camera parameters. This often leads to the problem of underfitting, as the simplified models lack the complexity to fully capture the information contained in these datasets. As a result, the models struggle to accurately reproduce the scene in novel views, leading to further artifacts and inconsistencies in the rendered graphics.

SUMMARY

[0006] Aspects and advantages of embodiments of the present disclosure will be set forth in part in the following description, or can be learned from the description, or can be learned through practice of the embodiments.

[0007] One example aspect of the present disclosure is directed to a computer-implemented method to train neural radiance fields. The method includes obtaining, by a computing system comprising one or more computing devices, a teacher neural radiance field model and a student neural radiance field model, wherein the teacher neural radiance field model has been previously trained to model a three-dimensional scene. The method includes generating, by the computing system, a teacher prediction for rendering parameters as an output of the teacher neural radiance field model, wherein the teacher prediction comprises a first predicted color for a synthetic pixel. The method includes generating, by the computing system, a student prediction for the rendering parameters as an output of the student neural radiance field model, wherein the student prediction comprises a second predicted color for the synthetic pixel. The method includes evaluating, by the computing system, a loss function that compares the first predicted color predicted by the teacher neural radiance field model with the second predicted color predicted by the student neural radiance field model. The method includes modifying, by the computing system, one or more values of one or more parameters of the student neural radiance field model based on the loss function.

[0008] Other aspects of the present disclosure are directed to various systems, apparatuses, non-transitory computer-readable media, user interfaces, and electronic devices.

[0009] These and other features, aspects, and advantages of various embodiments of the present disclosure will become better understood with reference to the following description and appended claims. The accompanying drawings, which are incorporated in and constitute a part of this specification, illustrate example embodiments of the present disclosure and, together with the description, serve to explain the related principles.

BRIEF DESCRIPTION OF THE DRAWINGS

[0010] Detailed discussion of embodiments directed to one of ordinary skill in the art is set forth in the specification, which makes reference to the appended figures, in which:

[0011] Figure 1 depicts a flow chart diagram of an example method for training neural radiance fields according to example embodiments of the present disclosure.

[0012] Figure 2 depicts a block diagram of an example computing system according to example embodiments of the present disclosure.

[0013] Figure 3A depicts a block diagram of an example computing device according to example embodiments of the present disclosure.

[0014] Figure 3B depicts a block diagram of an example computing device according to example embodiments of the present disclosure.

[0015] Figures 4A-C depict graphical diagrams of example coordinate systems according to example embodiments of the present disclosure.

[0016] Figure 5 depicts a graphical diagram of example teacher supervision according to example embodiments of the present disclosure.

[0017] Figure 6 depicts a graphical diagram of example ray jittering according to example embodiments of the present disclosure.

[0018] Reference numerals that are repeated across plural figures are intended to identify the same features in various implementations.

DETAILED DESCRIPTION

Overview

[0019] Generally, the present disclosure is directed to systems and methods to train neural radiance fields for improved rendering latency. The proposed framework can include the use of a ‘teacher’ model and a ‘student’ model. The teacher model is a neural radiance field model that has been previously trained to model a three-dimensional scene. The student model, on the other hand, is a neural radiance field model that is yet to be trained. This method is designed to improve the training of neural radiance fields for real-time rendering, which is a significant challenge in the field of computer graphics.

[0020] The process begins with the computing system generating a teacher prediction for rendering parameters. This prediction is the output of the teacher neural radiance field model and can include a first predicted color for a synthetic pixel. For example, if the three-dimensional scene being modeled is a room, the teacher model might predict that a certain

pixel in the rendering of the room should be a particular shade of blue. This prediction is based on the training that the teacher model has already received.

[0021] Next, the computing system generates a student prediction for the same rendering parameters. This prediction is the output of the student neural radiance field model and can include a second predicted color for the synthetic pixel. For instance, the student model might predict that the same pixel in the rendering of the room should be a slightly different shade of blue. This prediction is based on the current state of the student model, which has not yet been fully trained.

[0022] The computing system then evaluates a loss function that compares the first predicted color from the teacher model with the second predicted color from the student model. For example, the loss function can quantify the difference between the two predictions. For example, the loss function might calculate the absolute difference between the RGB values of the two predicted colors. The goal of this step is to assess how well the student model is currently performing compared to the teacher model.

[0023] Based on the evaluation of the loss function, the computing system modifies one or more values of one or more parameters of the student neural radiance field model. This modification is designed to improve the performance of the student model. For example, the computing system might adjust parameter values in the student model to reduce the difference between the student's predictions and the teacher's predictions.

[0024] In some implementations, generating the teacher prediction also involves generating teacher-predicted volume rendering weights. Similarly, generating the student prediction also involves generating student-predicted volume rendering weights. In some of such implementations, the loss function can further include a loss term that compares these teacher-predicted and student-predicted volume rendering weights. This allows the training process to take into account not only the predicted colors of the pixels but also the predicted volume rendering weights, which can influence the final appearance of the rendered scene.

[0025] In some implementations, the teacher prediction also includes teacher-predicted ray interval proposals. The student prediction can adopt or be based on these teacher-predicted ray interval proposals. This means that the student model learns to model the scene based on the teacher's understanding of scene geometry, which can help to improve the performance of the student model with respect to understanding scene geometry. Further, once trained, the student model may be structured so that is does not perform ray interval proposals (e.g., instead the student model may use a fixed sampling approach that does not require proposals). In this case, by having the student simply adopt or operate on the basis of

the teacher's ray interval proposals, a student ray interval proposal network is not unnecessarily trained.

[0026] Furthermore, in some implementations, the rendering parameters is a jittered camera parameters. This jittered location can be generated by adding random noise (e.g., from within a set distribution) to a ground truth camera parameters associated with a ground truth image that is included in the dataset on which the teacher model was trained. This technique, known as jittering, can help to increase the effective size of the training dataset and improve the performance of the student model.

[0027] In some implementations, the student neural radiance field model can comprise a plurality of neural radiance submodels. Each submodel can be associated with a different region of the three-dimensional scene. This allows the student model to specialize in different parts of the scene, which can help to improve the accuracy of the model and greatly improves the rendering latency of the smaller, student submodels.

[0028] In some implementations, the student model is parameterized using a set of spatially-structured parameter values. A respective subset of these parameter values is associated with each region of the scene. At runtime, the student model can adopt or otherwise operating using an interpolated set of parameter values generated by interpolating the subset of parameter values associated with the region that contains the rendering parameters. This allows the student model to adapt its parameters to the specific part of the scene that it is currently rendering, which can help to improve the quality of the rendered image.

[0029] The proposed techniques described in the patent application are designed to tackle the technical problem of latency in the rendering of neural radiance fields (NeRF), particularly in real-time applications. The present disclosure provides a technical solution to this problem by devising a novel training pipeline that distills a state-of-the-art 'teacher' neural radiance field model into a 'student' radiance field model that is better suited for real-time rendering.

[0030] One technical aspect of this present disclosure is the use of a 'teacher' model and a 'student' model. The 'teacher' model, which has already been trained, provides a reference for the 'student' model that is undergoing training. By comparing the predictions of the two models and adjusting the parameters of the student model based on this comparison, the present disclosure provides a method for improving the student model's performance. This technique of using a 'teacher' model to guide the training of a 'student' model is a technical solution that addresses the problem of underfitting in existing real-time NeRF models.

[0031] Another technical aspect of the present disclosure is the use of ray-jittering. By adding random noise to the camera rays before querying the teacher model, the effective size of the training dataset is increased. This technique improves the performance of the student model and is a technical solution to the problem of data scarcity in the training of real-time NeRF models.

[0032] With reference now to the Figures, example embodiments of the present disclosure will be discussed in further detail.

Example Methods

[0033] Figure 1 depicts a flow chart diagram of an example method for training neural radiance fields according to example embodiments of the present disclosure. The method begins at step 12, where a computing system obtains a teacher neural radiance field model and a student neural radiance field model. The teacher model has been previously trained to model a three-dimensional scene. The training data for the teacher model can include a set of 2D images of the scene, along with corresponding camera parameters. The student model, on the other hand, is an untrained or partially trained model that is being trained to model the same three-dimensional scene. The student model can have the same architecture as the teacher model or a different architecture, depending on the specific implementation.

[0034] At step 14, the computing system generates a teacher prediction for rendering parameters in the three-dimensional scene. The rendering parameters can be specified, for example, by a set of coordinates in a three-dimensional coordinate system and also a viewing direction. The teacher prediction is an output of the teacher neural radiance field model and includes a first predicted color for a synthetic pixel. The first predicted color can be an RGB value, for example, that corresponds to the color that the teacher model predicts for the synthetic pixel from the rendering parameters.

[0035] At step 16, the computing system generates a student prediction for the same rendering parameters. The student prediction is an output of the student neural radiance field model and includes a second predicted color for the synthetic pixel. The second predicted color can also be an RGB value, for instance, that corresponds to the color that the student model predicts for the synthetic pixel for the rendering parameters.

[0036] At step 18, the computing system evaluates a loss function that compares the first predicted color from the teacher model with the second predicted color from the student model. The loss function can be, for example, a mean squared error function or another suitable function that quantifies the difference between the two predicted colors. The loss

function serves to quantify the discrepancy between the predictions of the teacher model and the student model, thereby providing a measure of the current performance of the student model.

[0037] At step 20, the computing system modifies one or more values of one or more parameters of the student neural radiance field model based on the loss function. The parameters of the student model can include, for example, the weights and biases of a neural network used in the model and/or other forms of model parameters. The modification of the parameter values serves to improve the performance of the student model, with the goal of minimizing the discrepancy between the student model's predictions and the teacher model's predictions.

[0038] In some implementations of the present disclosure, the teacher prediction generated at step 14 also includes teacher-predicted volume rendering weights. Similarly, the student prediction generated at step 16 also includes student-predicted volume rendering weights. The volume rendering weights can be, for example, values that quantify the contribution of different parts of the scene to the final rendered image. The loss function evaluated at step 18 can further include a loss term that compares the teacher-predicted volume rendering weights to the student-predicted volume rendering weights. This additional loss term serves to further align the student model's predictions with the teacher model's predictions, thereby improving the performance of the student model.

[0039] In some implementations of the present disclosure, the teacher prediction generated at step 14 also includes teacher-predicted ray interval proposals. The student prediction generated at step 16 can be based on these teacher-predicted ray interval proposals, such that the student model learns to model the scene based on the teacher's understanding of scene geometry. This can help to improve the performance of the student model with respect to understanding scene geometry.

[0040] In some implementations of the present disclosure, the rendering parameters at which the teacher and student predictions are generated are jittered camera parameters. The jittered camera parameters can be generated by adding random noise to ground truth camera parameters associated with a ground truth image contained in the dataset on which the teacher neural radiance field was trained. This technique, known as jittering, can help to increase the effective size of the training dataset and improve the performance of the student model.

[0041] In some implementations of the present disclosure, the student neural radiance field model comprises a plurality of neural radiance submodels, each associated with a different region of the three-dimensional scene. This allows each student submodel to

specialize in different parts of the scene, which can help to improve the accuracy of the model and greatly improves the rendering latency of the smaller, student submodels.

[0042] In some implementations of the present disclosure, the student neural radiance field model is parameterized using a set of spatially-structured parameter values. A respective subset of these parameter values is associated with each of a plurality of regions of the three-dimensional scene. At runtime, the student neural radiance field model can adopt an interpolated set of parameter values generated by interpolating the respective subset of the spatially-structured parameter values associated with the region that contains the rendering parameters. This allows the student model to adapt its parameters to the specific part of the scene that it is currently rendering, which can help to improve the quality of the rendered image.

[0043] The method illustrated in Figure 1 provides a novel approach to training neural radiance field models for real-time rendering. By distilling the knowledge of a previously trained teacher model into a student model that may be smaller or may be parameterized to facilitate efficient rendering, the method can improve the performance of the student model and reduce the amount of computation required at inference time. This can lead to more accurate and faster rendering of three-dimensional scenes, thereby enhancing the user experience in applications such as virtual reality and augmented reality.

Example Devices and Systems

[0044] Figure 2 depicts a block diagram of an example computing system 100 according to example embodiments of the present disclosure. The system 100 includes a user computing device 102, a server computing system 130, and a training computing system 150 that are communicatively coupled over a network 180.

[0045] The user computing device 102 can be any type of computing device, such as, for example, a personal computing device (e.g., laptop or desktop), a mobile computing device (e.g., smartphone or tablet), a gaming console or controller, a wearable computing device, an embedded computing device, or any other type of computing device.

[0046] The user computing device 102 includes one or more processors 112 and a memory 114. The one or more processors 112 can be any suitable processing device (e.g., a processor core, a microprocessor, an ASIC, an FPGA, a controller, a microcontroller, etc.) and can be one processor or a plurality of processors that are operatively connected. The memory 114 can include one or more non-transitory computer-readable storage media, such as RAM, ROM, EEPROM, EPROM, flash memory devices, magnetic disks, etc., and

combinations thereof. The memory 114 can store data 116 and instructions 118 which are executed by the processor 112 to cause the user computing device 102 to perform operations.

[0047] In some implementations, the user computing device 102 can store or include one or more machine-learned models 120. For example, the machine-learned models 120 can be or can otherwise include various machine-learned models such as neural networks (e.g., deep neural networks) or other types of machine-learned models, including non-linear models and/or linear models. Neural networks can include feed-forward neural networks, recurrent neural networks (e.g., long short-term memory recurrent neural networks), convolutional neural networks or other forms of neural networks. Some example machine-learned models can leverage an attention mechanism such as self-attention. For example, some example machine-learned models can include multi-headed self-attention models (e.g., transformer models). Example machine-learned models 120 are discussed with reference to Figure 1.

[0048] In some implementations, the one or more machine-learned models 120 can be received from the server computing system 130 over network 180, stored in the user computing device memory 114, and then used or otherwise implemented by the one or more processors 112. In some implementations, the user computing device 102 can implement multiple parallel instances of a single machine-learned model 120 (e.g., to perform parallel neural rendering across multiple instances of renderers).

[0049] Additionally or alternatively, one or more machine-learned models 140 can be included in or otherwise stored and implemented by the server computing system 130 that communicates with the user computing device 102 according to a client-server relationship. For example, the machine-learned models 140 can be implemented by the server computing system 140 as a portion of a web service (e.g., a neural rendering service). Thus, one or more models 120 can be stored and implemented at the user computing device 102 and/or one or more models 140 can be stored and implemented at the server computing system 130.

[0050] The user computing device 102 can also include one or more user input components 122 that receives user input. For example, the user input component 122 can be a touch-sensitive component (e.g., a touch-sensitive display screen or a touch pad) that is sensitive to the touch of a user input object (e.g., a finger or a stylus). The touch-sensitive component can serve to implement a virtual keyboard. Other example user input components include a microphone, a traditional keyboard, or other means by which a user can provide user input.

[0051] The server computing system 130 includes one or more processors 132 and a memory 134. The one or more processors 132 can be any suitable processing device (e.g., a

processor core, a microprocessor, an ASIC, an FPGA, a controller, a microcontroller, etc.) and can be one processor or a plurality of processors that are operatively connected. The memory 134 can include one or more non-transitory computer-readable storage media, such as RAM, ROM, EEPROM, EPROM, flash memory devices, magnetic disks, etc., and combinations thereof. The memory 134 can store data 136 and instructions 138 which are executed by the processor 132 to cause the server computing system 130 to perform operations.

[0052] In some implementations, the server computing system 130 includes or is otherwise implemented by one or more server computing devices. In instances in which the server computing system 130 includes plural server computing devices, such server computing devices can operate according to sequential computing architectures, parallel computing architectures, or some combination thereof.

[0053] As described above, the server computing system 130 can store or otherwise include one or more machine-learned models 140. For example, the models 140 can be or can otherwise include various machine-learned models. Example machine-learned models include neural networks or other multi-layer non-linear models. Example neural networks include feed forward neural networks, deep neural networks, recurrent neural networks, and convolutional neural networks. Some example machine-learned models can leverage an attention mechanism such as self-attention. For example, some example machine-learned models can include multi-headed self-attention models (e.g., transformer models). Example models 140 are discussed with reference to Figure 1.

[0054] The user computing device 102 and/or the server computing system 130 can train the models 120 and/or 140 via interaction with the training computing system 150 that is communicatively coupled over the network 180. The training computing system 150 can be separate from the server computing system 130 or can be a portion of the server computing system 130.

[0055] The training computing system 150 includes one or more processors 152 and a memory 154. The one or more processors 152 can be any suitable processing device (e.g., a processor core, a microprocessor, an ASIC, an FPGA, a controller, a microcontroller, etc.) and can be one processor or a plurality of processors that are operatively connected. The memory 154 can include one or more non-transitory computer-readable storage media, such as RAM, ROM, EEPROM, EPROM, flash memory devices, magnetic disks, etc., and combinations thereof. The memory 154 can store data 156 and instructions 158 which are executed by the processor 152 to cause the training computing system 150 to perform

operations. In some implementations, the training computing system 150 includes or is otherwise implemented by one or more server computing devices.

[0056] The training computing system 150 can include a model trainer 160 that trains the machine-learned models 120 and/or 140 stored at the user computing device 102 and/or the server computing system 130 using various training or learning techniques, such as, for example, backwards propagation of errors. For example, a loss function can be backpropagated through the model(s) to update one or more parameters of the model(s) (e.g., based on a gradient of the loss function). Various loss functions can be used such as mean squared error, likelihood loss, cross entropy loss, hinge loss, and/or various other loss functions. Gradient descent techniques can be used to iteratively update the parameters over a number of training iterations.

[0057] In some implementations, performing backwards propagation of errors can include performing truncated backpropagation through time. The model trainer 160 can perform a number of generalization techniques (e.g., weight decays, dropouts, etc.) to improve the generalization capability of the models being trained.

[0058] In particular, the model trainer 160 can train the machine-learned models 120 and/or 140 based on a set of training data 162. The training data 162 can include, for example, multiple images of a scene.

[0059] In some implementations, if the user has provided consent, the training examples can be provided by the user computing device 102. Thus, in such implementations, the model 120 provided to the user computing device 102 can be trained by the training computing system 150 on user-specific data received from the user computing device 102. In some instances, this process can be referred to as personalizing the model.

[0060] The model trainer 160 includes computer logic utilized to provide desired functionality. The model trainer 160 can be implemented in hardware, firmware, and/or software controlling a general purpose processor. For example, in some implementations, the model trainer 160 includes program files stored on a storage device, loaded into a memory and executed by one or more processors. In other implementations, the model trainer 160 includes one or more sets of computer-executable instructions that are stored in a tangible computer-readable storage medium such as RAM, hard disk, or optical or magnetic media.

[0061] The network 180 can be any type of communications network, such as a local area network (e.g., intranet), wide area network (e.g., Internet), or some combination thereof and can include any number of wired or wireless links. In general, communication over the network 180 can be carried via any type of wired and/or wireless connection, using a wide

variety of communication protocols (e.g., TCP/IP, HTTP, SMTP, FTP), encodings or formats (e.g., HTML, XML), and/or protection schemes (e.g., VPN, secure HTTP, SSL).

[0062] Figure 2 illustrates one example computing system that can be used to implement the present disclosure. Other computing systems can be used as well. For example, in some implementations, the user computing device 102 can include the model trainer 160 and the training dataset 162. In such implementations, the models 120 can be both trained and used locally at the user computing device 102. In some of such implementations, the user computing device 102 can implement the model trainer 160 to personalize the models 120 based on user-specific data.

[0063] Figure 3A depicts a block diagram of an example computing device 10 that performs according to example embodiments of the present disclosure. The computing device 10 can be a user computing device or a server computing device.

[0064] The computing device 10 includes a number of applications (e.g., applications 1 through N). Each application contains its own machine learning library and machine-learned model(s). For example, each application can include a machine-learned model. Example applications include a text messaging application, an email application, a dictation application, a virtual keyboard application, a browser application, etc.

[0065] As illustrated in Figure 3A, each application can communicate with a number of other components of the computing device, such as, for example, one or more sensors, a context manager, a device state component, and/or additional components. In some implementations, each application can communicate with each device component using an API (e.g., a public API). In some implementations, the API used by each application is specific to that application.

[0066] Figure 3B depicts a block diagram of an example computing device 50 that performs according to example embodiments of the present disclosure. The computing device 50 can be a user computing device or a server computing device.

[0067] The computing device 50 includes a number of applications (e.g., applications 1 through N). Each application is in communication with a central intelligence layer. Example applications include a text messaging application, an email application, a dictation application, a virtual keyboard application, a browser application, etc. In some implementations, each application can communicate with the central intelligence layer (and model(s) stored therein) using an API (e.g., a common API across all applications).

[0068] The central intelligence layer includes a number of machine-learned models. For example, as illustrated in Figure 3B, a respective machine-learned model can be provided for

each application and managed by the central intelligence layer. In other implementations, two or more applications can share a single machine-learned model. For example, in some implementations, the central intelligence layer can provide a single model for all of the applications. In some implementations, the central intelligence layer is included within or otherwise implemented by an operating system of the computing device 50.

[0069] The central intelligence layer can communicate with a central device data layer. The central device data layer can be a centralized repository of data for the computing device 50. As illustrated in Figure 3B, the central device data layer can communicate with a number of other components of the computing device, such as, for example, one or more sensors, a context manager, a device state component, and/or additional components. In some implementations, the central device data layer can communicate with each device component using an API (e.g., a private API).

Example Memory-Efficient Radiance Fields

[0070] This section includes a review of MERF (Reiser et al., *MERF: Memory-Efficient Radiance Fields for Real-time View Synthesis in Unbounded Scenes*, arXiv:2302.12249 (2023)). MERF maps 3D positions $\mathbf{x} \in \mathbb{R}^3$ to feature vectors $\mathbf{t} \in \mathbb{R}^8$. MERF parameterizes this mapping using a combination of high-resolution triplanes ($\mathbf{P}_x$, $\mathbf{P}_y$, $\mathbf{P}_z \in \mathbb{R}^{R\times R\times 8}$) and a low-resolution sparse voxel grid $\mathbf{V} \in \mathbb{R}^{L\times L\times L\times 8}$. A query point $\mathbf{x}$ is projected onto each of three axis-aligned planes and the underlying 2D grid is queried via bilinear interpolation. Additionally, a trilinear sample is taken from the sparse voxel grid. The resulting four 8-dimensional vectors are then summed:

$$\mathbf{t}_{\text{MERF}}(\mathbf{x}) = \mathbf{P}_x(\mathbf{x}) + \mathbf{P}_y(\mathbf{x}) + \mathbf{P}_z(\mathbf{x}) + \mathbf{V}(\mathbf{x}). \quad (1)$$

[0071] This vector is then unpacked into three parts, which are independently rectified to yield a scalar density, a diffuse RGB color, and a feature vector that encodes view-dependence effects:

$$\tau = \exp(\mathbf{t}_1), \mathbf{c} = sigmoid(\mathbf{t}_{2:4}), \mathbf{f} = sigmoid(\mathbf{t}_{5:8}). \quad (2)$$

[0072] To render a pixel, a ray is cast from that pixel's center of projection $\mathbf{o}$ along the viewing direction $\mathbf{d}$ and sampled at a set of distances $\{t_i\}$ to generate a set of points along that ray $\mathbf{x}_i = \mathbf{o} + t_i\mathbf{d}$. The densities $\{\tau_i\}$ are converted into alpha compositing weights $\{w_i\}$ using the numerical quadrature approximation for volume rendering [?]:

$$w_i = T_i\big(1 - \exp(-\tau_i\delta_i)\big), where\ T_i = \exp\left(-\sum_{j<i} \tau_j\delta_j\right), \quad (3)$$

where $\delta_i = t_{i+1} - t_i$ is the distance between adjacent samples. After alpha compositing, a deferred shading approach is used to decode the blended diffuse RGB colors $\sum_i w_i \mathbf{c}_i$ and the blended view-dependent color features $\sum_i w_i \mathbf{f}_i$ into the final pixel color with the help of the small, deferred rendering MLP $h(\cdot;\theta)$:

$$\left(\sum_i w_i \mathbf{c}_i\right) + h\left(\sum_i w_i \mathbf{c}_i, \sum_i w_i \mathbf{f}_i, \mathbf{d}; \theta\right), \quad (4)$$

where θ are the MLP's parameters.

[0073] In unbounded scenes, far-away content can be modelled coarsely. To achieve a resolution that drops off with the distance from the scene's focus point, MERF applies a contraction function to each spatial position $\mathbf{x}$ before querying the feature field:

$$contract(\mathbf{x})_d = \begin{cases} x_d & \text{if } \| \mathbf{x} \|_\infty \leq 1 \\ \dfrac{x_d}{\| \mathbf{x} \|_\infty} & \text{if} x_d \neq \| \mathbf{x} \|_\infty > 1 \\ \left(2 - \dfrac{1}{|x_d|}\right)\dfrac{x_d}{|x_d|} & \text{if} x_d = \| \mathbf{x} \|_\infty > 1 \end{cases} \quad (5)$$

Example Streamable Memory Efficient Radiance Fields

[0074] Although real-time view-synthesis methods like MERF perform well for a localized environment, they often fail to scale to large multi-room scenes. To this end, some example implementations of the present disclosure leverage a hierarchical architecture. First, some example implementations partition the coordinate space of the entire scene into a series of blocks, where each block is modeled by its own MERF -like representation. Second, some example implementations introduce a grid of spatially-anchored network parameters within each block for modeling view-dependent effects. Finally, some example implementations introduce a gating mechanism for modulating high- and low-resolution contributions to a location's feature representation. Thus, one example overall architecture can be thought of as a three-level hierarchy: based on the camera origin, (i) some example implementations select an appropriate submodel, then within a submodel (ii) some example implementations compute the parameters of a deferred appearance network via interpolation, and then within a local voxel neighborhood, (iii) some example implementations determine a location's feature representation via feature gating.

[0075] This greatly increases the capacity of the proposed model without diminishing rendering speed or increasing memory consumption: even as total storage requirements increase with the number of submodels, only a single submodel is needed to render a given

frame. As such, when implemented on a graphics accelerator, the proposed system maintains modest resource requirements comparable to MERF .

[0076] Example Coordinate Space Partitioning Techniques

[0077] While MERF offers sufficient capacity for faithfully representing medium-scale scenes, the use of a single set of triplanes limits its capacity and reduces image quality. In large scenes, numerous surface points project to the same 2D plane location, and the representation therefore struggles to simultaneously represent high-frequency details of multiple surfaces. Although this can be partially ameliorated by increasing the spatial resolution of the underlying representation, doing so significantly impacts memory consumption and is prohibitively expensive in practice.

[0078] Therefore, some example implementations of the present disclosure instead coarsely subdivide the scene into a 3D grid based on camera origin and associate each grid cell with an independent submodel. Each submodel is assigned its own contraction space (Eq. (5)) and is tasked with representing the region of the scene within its grid cell at high detail, while the region outside each submodel's cell is modeled coarsely. Note that, typically, the entire scene is still represented by each submodel — the submodels differ only in terms of which portions of the scene lies inside or outside of each submodel's contraction region. As such, rendering a camera only requires a single submodel, implying that only one submodel must be in memory at a time.

[0079] Formally, some example implementations shift and scale all training cameras to lie within a $[-\mathrm{K}/2, \mathrm{K}/2]^3$ cube, and then partition this cube into K^3 identical and tightly packed subvolumes of size $[-1,1]^3$. Some example implementations can assign training cameras to submodels $\{\mathcal{S}_k\}$ by identifying the associated subvolume $\mathbf{R}_k$ that each camera origin $\mathbf{o}$ lies within:

$$s^* = arg\ min_{k\in\{1..\mathrm{K}^3\}} \mathrm{D}(\mathbf{o}, k), \text{where } \mathrm{D}(\mathbf{o}, k) = \min_{\mathbf{x}\in\mathbf{R}_k} ||\mathbf{o} - \mathbf{x}||_\infty. \quad (6)$$

[0080] Some example implementations configure the proposed camera-to-submodel assignment procedure to apply to cameras outside of the training set, ensuring its validity during test set rendering. This enables a wide range of features including ray jittering, submodel reassignment, a submodel consensus loss, arbitrary test camera placement, and ping-pong buffers.

[0081] Rather than exhaustively instantiating submodels for all K^3 subvolumes, some example implementations only consider subvolumes that contain at least one training camera.

As most scenes are outdoors or single-story buildings, this reduces the number of submodels from K^3 to $O(K^2)$.

[0082] Example Deferred Appearance Network Partitioning Techniques

[0083] The second level in the proposed partitioning hierarchy concerns the deferred rendering model. Recall that MERF employs a small multi-layer perceptron (MLP) to decode view-dependent colors from blended features as described in Eq. (4). Although the small size of this network is advantageous for fast inference, in some specific cases its capacity is insufficient to accurately reproduce complex view-dependent effects common in larger scenes. Simply increasing the size of this network is not viable as doing so would significantly reduce rendering speed.

[0084] Instead, some example implementations uniformly subdivide the domain of each submodel into a lattice with P vertices along each axis. Some example implementations associate each cell $(u, v, w) \in \{1, .., P\}^3$ with a separate set of network parameters θ_{uvw} and trilinearly interpolate them based on camera origin $\mathbf{o}$:

$$\theta = Trilerp(\mathbf{o}, \{\theta_{uvw} : (u, v, w) \in \{1, .., P\}^3\}) \quad (7)$$

[0085] The use of trilinear interpolation, unlike the nearest-neighbor interpolation used in coordinate space partitioning, helps to prevent aliasing of the view-dependent MLP parameters, which takes the form of conspicuous "popping" artifacts in specular highlights as the camera moves through space. Some example implementations further reduce popping between submodels via regularization. After parameter interpolation, view-dependent colors can be decoded according to Eq. (4).

[0086] Since the size of the view-dependent MLP is negligible compared to total representation size, deferred network partitioning has almost no effect on memory consumption or storage impact. As a result, this technique increases model capacity almost for free. This is in contrast to coordinate space partitioning, which significantly increases storage size. The union of coordinate space and deferred network partitioning effectively increases spatial and view-dependent resolution, respectively. See Figures 4A-C for an illustration of these two forms of partitioning.

[0087] Specifically, Figures 4A-C illustrate coordinate systems in example implementations of the present disclosure for a scene with $K^3 = 3^3$ coordinate space partitions and $P^3 = 4^3$ deferred appearance network sub-partitions. Each partition is capable of representing the entire scene while allocating the majority of its model capacity to its corresponding partition. Within each partition, some example implementations instantiate a

set of spatially-anchored MLP weights $\{\theta_{i,j}\}$ parameterizing the deferred appearance model, which some example implementations trilinearly interpolate as a function of the camera origin o during rendering. Specifically, Figure 4A illustrates the entire scene 402 in world coordinates with the scene partition 404 and two submodels 406 and 408. Figures 4B and 4C illustrate the same scene 402 from the view of two submodels 406 and 408 in their corresponding contracted coordinate systems. Specifically, Figure 4B visualizes the rendering and parameter interpolation process when the camera origin **o** 450 lies inside of a submodel's partition 452, and Figure 4C visualizes the rendering and parameter interpolation process when the camera origin **o** 460 lies inside of a submodel's partition 462.

[0088] Example Feature Gating Techniques

[0089] The final level in the proposed hierarchy is at the level of coarse 3D voxel grid $\mathbf{V}$ and three high-resolution planes ($\mathbf{P}_x$, $\mathbf{P}_y$, $\mathbf{P}_z$). In MERF, each 3D position is associated with an 8-dimensional feature vector: the sum of the contributions from these four sources (See Eq. (1)). Although effective, the features generated by this procedure are limited by their naive use of summation to merge high- and low-resolution information, entangling the two together.

[0090] In contrast, some example implementations of the present disclosure instead use low-resolution 3D features to "gate" high-resolution features: if high-resolution features add value for a given 3D coordinate, they should be employed; otherwise, they should be ignored and the smoother, low-resolution features should be used. To this end, some example implementations modify feature aggregation as follows: instead of a naive summation, some example implementations take the last component $w(\mathbf{x}) = [\mathbf{V}(\mathbf{x})]_8$ of the low-resolution voxel grid's contribution and use it to scale the triplane feature contributions:

$$\hat{\mathbf{t}}(\mathbf{x}) = w(\mathbf{x}) \cdot \Big(\mathbf{P}_x(\mathbf{x}) + \mathbf{P}_y(\mathbf{x}) + \mathbf{P}_z(\mathbf{x})\Big) + \mathbf{V}(\mathbf{x}). \quad (8)$$

[0091] Some example implementations then build the proposed final feature representation by concatenating the aggregated features $\hat{\mathbf{t}}(\mathbf{x})$ and the voxel grid features $\mathbf{V}(\mathbf{x})$:

$$\mathbf{t}(\mathbf{x}) = \hat{\mathbf{t}}(\mathbf{x}) \oplus \mathbf{V}(\mathbf{x}). \quad (9)$$

[0092] This incentivizes the model to leverage the low-resolution voxel grid to disable the high-resolution triplanes when rendering low-frequency content such as featureless white walls, and gives the model the freedom to focus on detailed parts of the scene. This change can also be thought of as a sort of "attention", as a multiplicative interaction is being used to determine when the model should "attend" to the triplane features. This change slightly

affects the memory and speed of the proposed model by virtue of increasing the number of rows in the first weight matrix of the proposed MLP, but the practical impact of this is negligible.

Example Training Techniques

[0093] Example Radiance Field Distillation Techniques

[0094] NeRF-like models such as MERF are traditionally trained "from scratch" to minimize photometric loss on a set of posed input images. Regularization is very beneficial when training such systems to improve generalization to novel views. One example employs a family of carefully tuned losses in addition to photometric reconstruction error to achieve state-of-the-art performance on large, multi-room scenes. Some example implementations instead adopt "student/teacher" distillation and train the proposed representation to imitate an already-trained, state-of-the-art radiance field model.

[0095] Distillation has several advantages: some example implementations inherit the helpful inductive biases of the teacher model, circumvent the need for onerous hyperparameter tuning for generalization, and enable the recovery of local representations that are also globally consistent. The proposed approach achieves quality comparable to its a larger teacher while being three orders of magnitude faster to render.

[0096] Some example implementations supervise the proposed model by distilling the appearance and geometry of a reference radiance field. Note that the teacher model is frozen during optimization. See Figure 5 for a visualization. Specifically, Figure 5 illustrates an example teacher supervision approach. The student receives photometric supervision via rendered colors and geometric supervision via the rendering weights along camera rays. Both models operate on the same set of ray intervals.

[0097] *Appearance*:

[0098] Some example implementations supervise the proposed model by minimizing the photometric difference between patches predicted by the proposed model and a source of ground-truth. Instead of limiting training to a set of photos representing a small subset of a scene's plenoptic function, the proposed source of "ground-truth" image patches is a teacher model rendered from an arbitrary set of cameras. Specifically, some example implementations distill appearance by penalizing the discrepancy between 3×3 patches rendered from student and teacher models. Some example implementations use a weighted

combination of the RMSE and DSSIM losses between each student patch $\mathbf{C}$ and its corresponding teacher patch $\mathbf{C}^*$:

$$\mathcal{L}_{\mathbf{c}} = 1.5 \cdot \text{DSSIM}(\mathbf{C}, \mathbf{C}^*) + \sum_{\mathbf{c} \in \mathbf{C}} \left|\left|\mathbf{c} - \mathbf{c}^*\right|\right|_2 . \quad (10)$$

[0099] *Geometry*:

[0100] To distill geometry, some example implementations begin by querying the proposed teacher with a given ray origin and direction. This yields a set of weighted intervals along the camera ray $\{((t_i, t_{i+1}), w_i^T)\}$, where each (t_i, t_{i+1}) are the metric distances along the ray corresponding to interval i, and each w_i^T is the teacher's corresponding alpha compositing weight for the same interval as per Eq. (3). The weight of each interval i reflects its contribution to the final predicted radiance. It is this quantity that some example implementations distill into the proposed student. Specifically, some example implementations compute the absolute difference between the teacher and student weights:

$$L_\tau = \sum_i \left| w_i^T - w_i^S \right| . \quad (11)$$

[0101] Because these volumetric rendering weights are a function of volumetric density (Eq. (3)), this loss on weights indirectly encourages the student's and teacher's density fields to be consistent with each other in visible regions of the scene.

[0102] Example Data Augmentation Techniques

[0103] Because the proposed distillation approach enables the supervision of the proposed student model at any ray in Euclidean space, some example implementations require a procedure for selecting a useful set of rays. While sampling rays uniformly at random throughout the scene is viable, this approach leads to poor reconstruction quality as many such rays originate from inside of objects or walls or are pointed towards unimportant or under-observed parts of the scene. Using camera rays corresponding to pixels in the dataset used to train the teacher model also performs poorly, as those input images represent a tiny subset of possible views of the scene. As such, some example implementations adopt a compromise approach by using randomly-perturbed versions of the dataset's camera rays, which yields a kind of "data augmentation" that improves generalization while focusing the student model's attention towards the parts of the scene that the photographer deemed relevant.

[0104] To generate a training ray, some example implementations first randomly select a ray from the teacher's dataset with origin $\mathbf{o}$ and direction $\mathbf{d}$. Some example implementations

then jitter the origin with isotropic Gaussian noise and draw a uniform sample from an ε-neighborhood of the ray's direction vector to obtain a ray $(\hat{\mathbf{o}}, \hat{\mathbf{d}})$:

$$\hat{\mathbf{o}} \sim \mathcal{N}(\mathbf{o}, \sigma^2 \mathbb{I}), \quad (12)$$

$$\hat{\mathbf{d}} \sim \mathcal{U}\left(\left\{\mathbf{v} \in \mathbb{R}^3 : \left|\left|\mathbf{v} - \mathbf{d}\right|\right|_2 < \varepsilon, \left|\left|\mathbf{v}\right|\right|_2 = 1\right\}\right). \quad (13)$$

[0105] Some example implementations set $\sigma = 0.03\text{K}$ and $\varepsilon = 0.03$. Note that σ is defined in normalized scene coordinates, where all the input cameras are contained within the $[-\text{K}/2, \text{K}/2]^3$ cube; See Figure 6 as an example. Specifically, Figure 6 illustrates an example ray jittering process. To generate training rays for the proposed student model (e.g., shown as the smaller cameras such as camera 602) some example implementations randomly perturb the origins and directions of the camera rays used to supervise the proposed teacher model (e.g., shown as the larger camera 604).

[0106] Example Submodel Consistency Techniques

[0107] Recall that, in spite of employing a single teacher model, coordinate space partitioning means that some example implementations are effectively training multiple, independent student submodels in parallel (in some cases, all submodels are actually trained simultaneously on a single host). This presents a challenge in terms of consistency across submodels — at test time, we seek temporal consistency under smooth camera motion, even when transitioning between submodels. This can be ameliorated by rendering multiple submodels and blending their results, but doing so significantly slows rendering and requires the presence of multiple submodels at once. In contrast, some example implementations aim to render each frame while only querying a single submodel.

[0108] To encourage adjacent submodels to make similar predictions for a given camera ray, some example implementations introduce a photometric consistency loss between submodels. During training some example implementations render each camera ray in the proposed batch twice: once using its "home" submodel s (whichever submodel the ray origin lies within the interior of), and again using a randomly-chosen neighboring submodel $\hat{s}$. Some example implementations then impose a straightforward loss between those two rendered colors:

$$\mathcal{L}_s = ||\mathbf{c}_s(\mathbf{r}) - \mathbf{c}_{\hat{s}}(\mathbf{r})||_2 . \quad (14)$$

[0109] Additionally, when constructing batches of training rays, some example implementations take care to assign each ray to a submodel where it will meaningfully improve reconstruction quality. Intuitively, some example implementations expect rays to add the most value to their "home" submodel, but the rays that originate from neighboring

submodels may also provide value by providing additional viewing angles of scene content within a submodel's interior. As such, some example implementations first assign each training ray its "home" submodel, and then randomly re-assign some percentage (e.g., 20%) of rays per batch to a randomly-selected adjacent submodel.

Example Rendering Techniques

[0110] Example Baking Techniques

[0111] After training, some example implementations generate precomputed assets for the real-time viewer. Some example implementations broadly follow the "baking" process of MERF with the changes described here and other minor modifications. Recall that MERF extracts a multi-resolution occupancy grid for two purposes: (i) to mark the parts in the scene for baking, and (ii) to skip empty space during rendering. Some example implementations eliminate spurious floaters by post-processing this occupancy grid (at the highest resolution $R^3 = 2048^3$) with a 3x3x3 median filter. As output, some example implementations produce an independent set of baked assets for each submodel, where each asset collection closely mirrors that produced by MERF. These assets include three high-resolution 2D feature maps and a sparse low-resolution 3D feature grid, both represented as quantized byte arrays. The deferred network parameters, in contrast, retain their floating point representation. Unlike MERF, some example implementations store assets as gzip-compressed binary blobs, which are slightly smaller and significantly faster to decode than PNG images.

[0112] Example Live Viewer

[0113] An example viewer proposed herein is based on the MERF volume renderer, i.e an OpenGL fragment shader that implements both ray marching and deferred rendering, using texture look-ups to retrieve pointwise feature representations and density. However, the proposed implementation can contain several modifications: support for submodels and parameter interpolation, a distance grid acceleration structure, and/or other low-level optimizations. The proposed viewer has modest compute and memory requirements, enabling real-time rendering on smartphones and other resource-constrained devices.

[0114] *Submodels*:

[0115] Recall that the proposed method only requires a single submodel to render any given viewpoint in the scene, strongly bounding peak memory usage. To hide network latency, some example implementations nevertheless "ping-pong" two submodels in and out of GPU memory as the user interactively explores the environment. When the camera enters a new subvolume, some example implementations begin loading the corresponding submodel

into CPU memory while the previous submodel, which includes an approximate representation of this new adjacent region, continues to be used for rendering. Once the new submodel is available, some example implementations transfer it to GPU memory and immediately use it for rendering. Peak GPU memory usage is thus limited to a maximum of two submodels: one actively being rendered and one being loaded into memory. Some example implementations further limit CPU memory usage by evicting loaded submodels from memory via least-recently-used caching.

[0116] *Deferred Appearance Network Interpolation*:

[0117] While trilinearly interpolating deferred appearance network parameters requires additional computation, this feature has a negligible impact on frame rendering time. As all pixels in an image share a common ray origin, some example implementations only need to interpolate parameters once per frame. In practice, some example implementations perform this interpolation on the CPU before executing the fragment shader.

Additional Disclosure

[0118] The technology discussed herein makes reference to servers, databases, software applications, and other computer-based systems, as well as actions taken and information sent to and from such systems. The inherent flexibility of computer-based systems allows for a great variety of possible configurations, combinations, and divisions of tasks and functionality between and among components. For instance, processes discussed herein can be implemented using a single device or component or multiple devices or components working in combination. Databases and applications can be implemented on a single system or distributed across multiple systems. Distributed components can operate sequentially or in parallel.

[0119] While the present subject matter has been described in detail with respect to various specific example embodiments thereof, each example is provided by way of explanation, not limitation of the disclosure. Those skilled in the art, upon attaining an understanding of the foregoing, can readily produce alterations to, variations of, and equivalents to such embodiments. Accordingly, the subject disclosure does not preclude inclusion of such modifications, variations and/or additions to the present subject matter as would be readily apparent to one of ordinary skill in the art. For instance, features illustrated or described as part of one embodiment can be used with another embodiment to yield a still further embodiment. Thus, it is intended that the present disclosure cover such alterations, variations, and equivalents.

WHAT IS CLAIMED IS:

1. A computer-implemented method to train neural radiance fields, the method comprising:

obtaining, by a computing system comprising one or more computing devices, a teacher neural radiance field model and a student neural radiance field model, wherein the teacher neural radiance field model has been previously trained to model a three-dimensional scene;

generating, by the computing system, a teacher prediction for rendering parameters as an output of the teacher neural radiance field model, wherein the teacher prediction comprises a first predicted color for a synthetic pixel;

generating, by the computing system, a student prediction for the rendering parameters as an output of the student neural radiance field model, wherein the student prediction comprises a second predicted color for the synthetic pixel;

evaluating, by the computing system, a loss function that compares the first predicted color predicted by the teacher neural radiance field model with the second predicted color predicted by the student neural radiance field model; and

modifying, by the computing system, one or more values of one or more parameters of the student neural radiance field model based on the loss function.

2. The computer-implemented method of any preceding claim, wherein:

generating, by the computing system, the teacher prediction comprises generating teacher-predicted volume rendering weights;

generating, by the computing system, the student prediction comprises generating student-predicted volume rendering weights; and

the loss function further comprises a loss term that compares the teacher-predicted volume rendering weights to the student-predicted volume rendering weights.

3. The computer-implemented method of any preceding claim, wherein:

generating, by the computing system, the teacher prediction comprises generating teacher-predicted ray interval proposals;

generating, by the computing system, the student prediction comprises adopting the teacher-predicted ray interval proposals for the student neural radiance field.

4. The computer-implemented method of any preceding claim, wherein the rendering parameters comprises a jittered camera parameters that has been generated by jittering a ground truth camera parameters associated with a ground truth image contained in a dataset on which the teacher neural radiance field was trained.

5. The computer-implemented method of claim 4, wherein the jittered camera parameters is not contained in the dataset on which the teacher neural radiance field was trained.

6. The computer-implemented method of any preceding claim, wherein the student neural radiance field model comprises a plurality of neural radiance submodels respectively associated with a plurality of regions of the three-dimensional scene.

7. The computer-implemented method of any preceding claim, wherein the student neural radiance field model is parameterized using a set of spatially-structured parameter values, wherein a respective subset of the spatially-structured parameter values is associated with each of a plurality of regions of the three-dimensional scene, and wherein, at runtime, the student neural radiance field model adopts an interpolated set of parameter values generated by interpolating the respective subset of the spatially-structured parameter values associated with the region that contains the rendering parameters.

8. One or more non-transitory computer-readable media that collectively store a student neural radiance field model trained according to the method of any preceding claim.

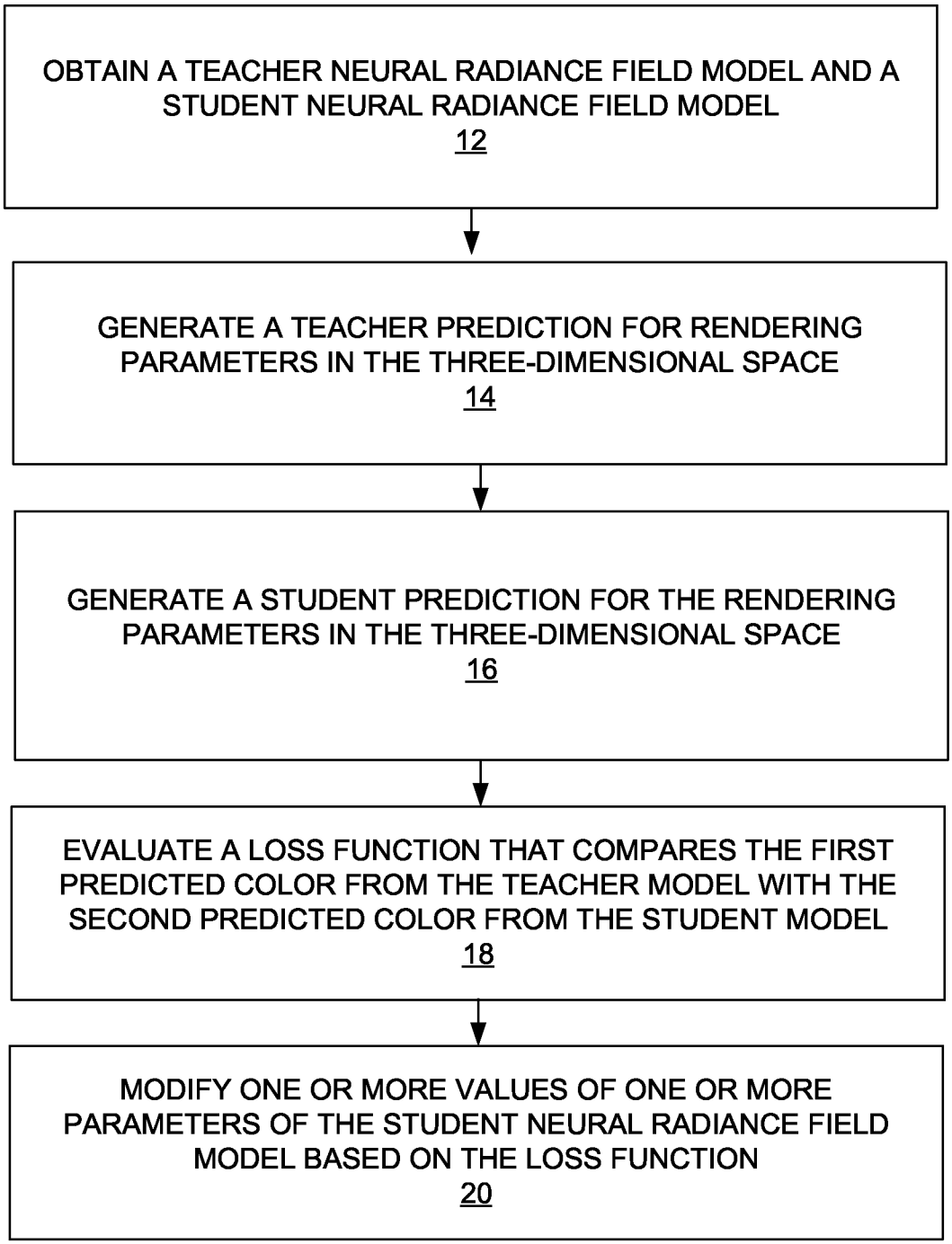

OBTAIN A TEACHER NEURAL RADIANCE FIELD MODEL AND A STUDENT NEURAL RADIANCE FIELD MODEL
12
GENERATE A TEACHER PREDICTION FOR RENDERING PARAMETERS IN THE THREE-DIMENSIONAL SPACE
14
GENERATE A STUDENT PREDICTION FOR THE RENDERING PARAMETERS IN THE THREE-DIMENSIONAL SPACE
16
EVALUATE A LOSS FUNCTION THAT COMPARES THE FIRST PREDICTED COLOR FROM THE TEACHER MODEL WITH THE SECOND PREDICTED COLOR FROM THE STUDENT MODEL
18
MODIFY ONE OR MORE VALUES OF ONE OR MORE PARAMETERS OF THE STUDENT NEURAL RADIANCE FIELD MODEL BASED ON THE LOSS FUNCTION
20

Figure 1

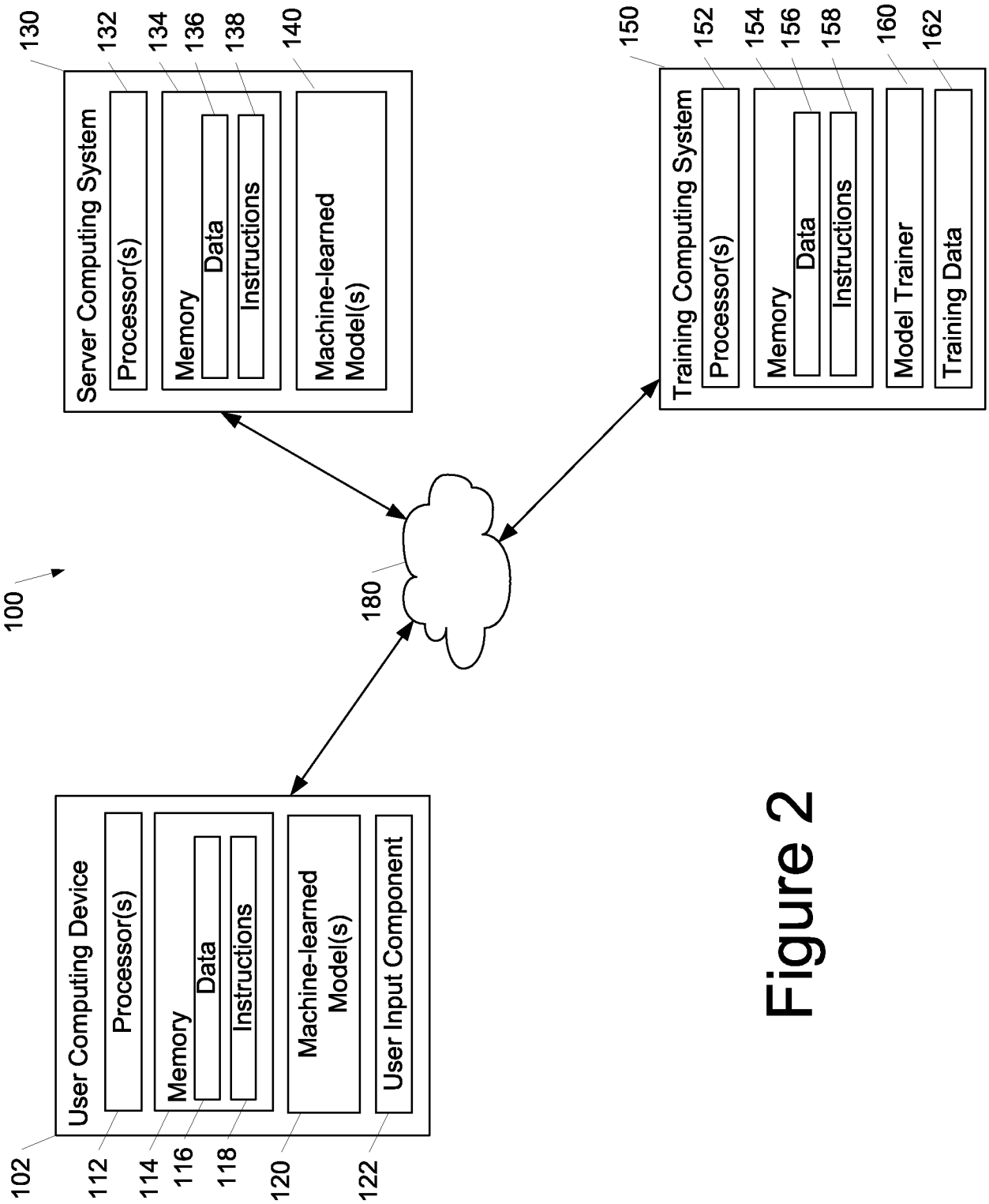

100
180
User Computing Device
Processor(s)
Memory
Data
Instructions
Machine-learned Model(s)
User Input Component
102
112
114
116
118
120
122
Server Computing System
Processor(s)
Memory
Data
Instructions
Machine-learned Model(s)
130
132
134
136
138
140
Training Computing System
Processor(s)
Memory
Data
Instructions
Model Trainer
Training Data
150
152
154
156
158
160
162

Figure 2

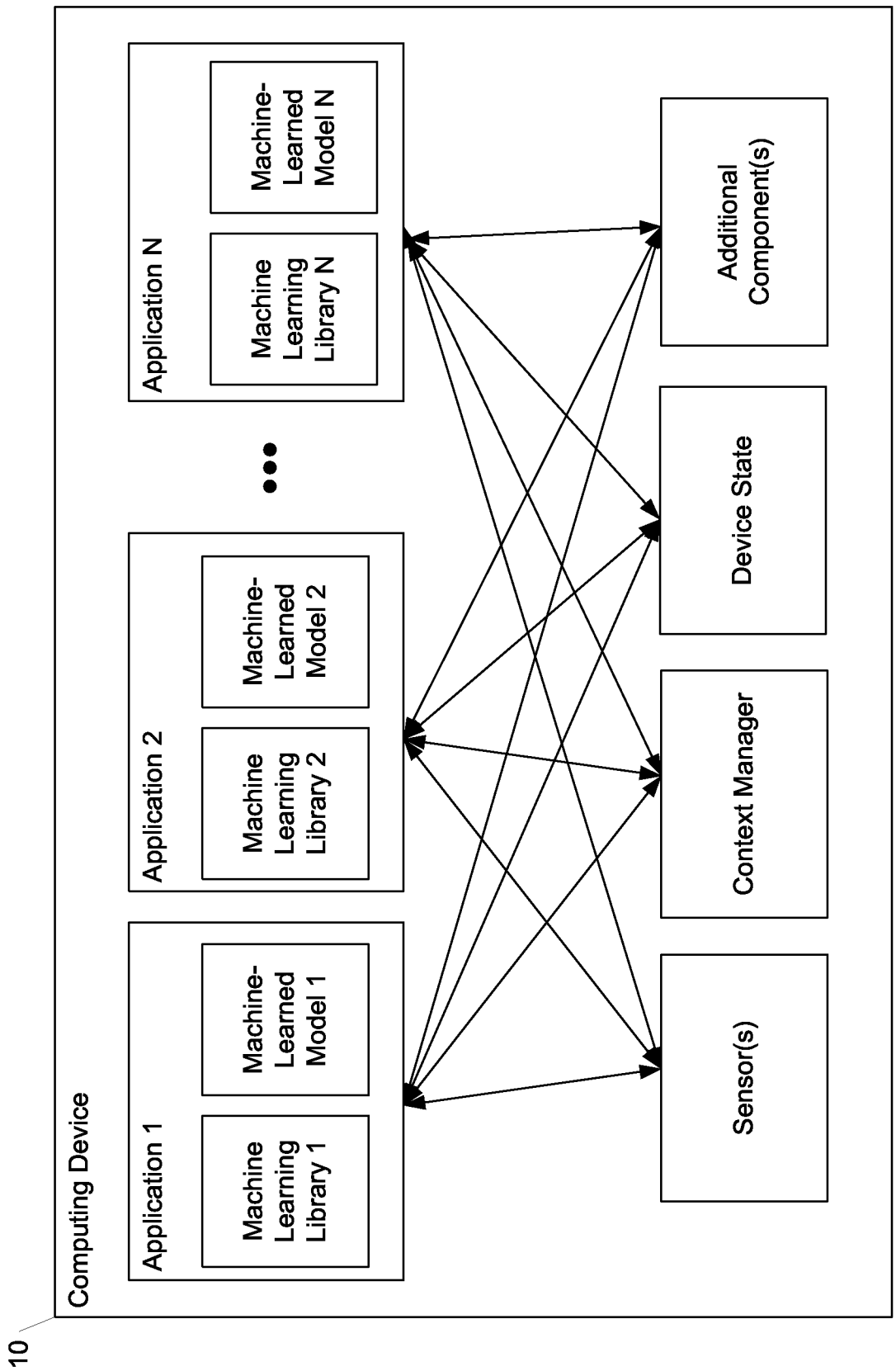

10
Computing Device
Application 1
Machine Learning Library 1
Machine-Learned Model 1
Application 2
Machine Learning Library 2
Machine-Learned Model 2
Application N
Machine Learning Library N
Machine-Learned Model N
Sensor(s)
Context Manager
Device State
Additional Component(s)

Figure 3A

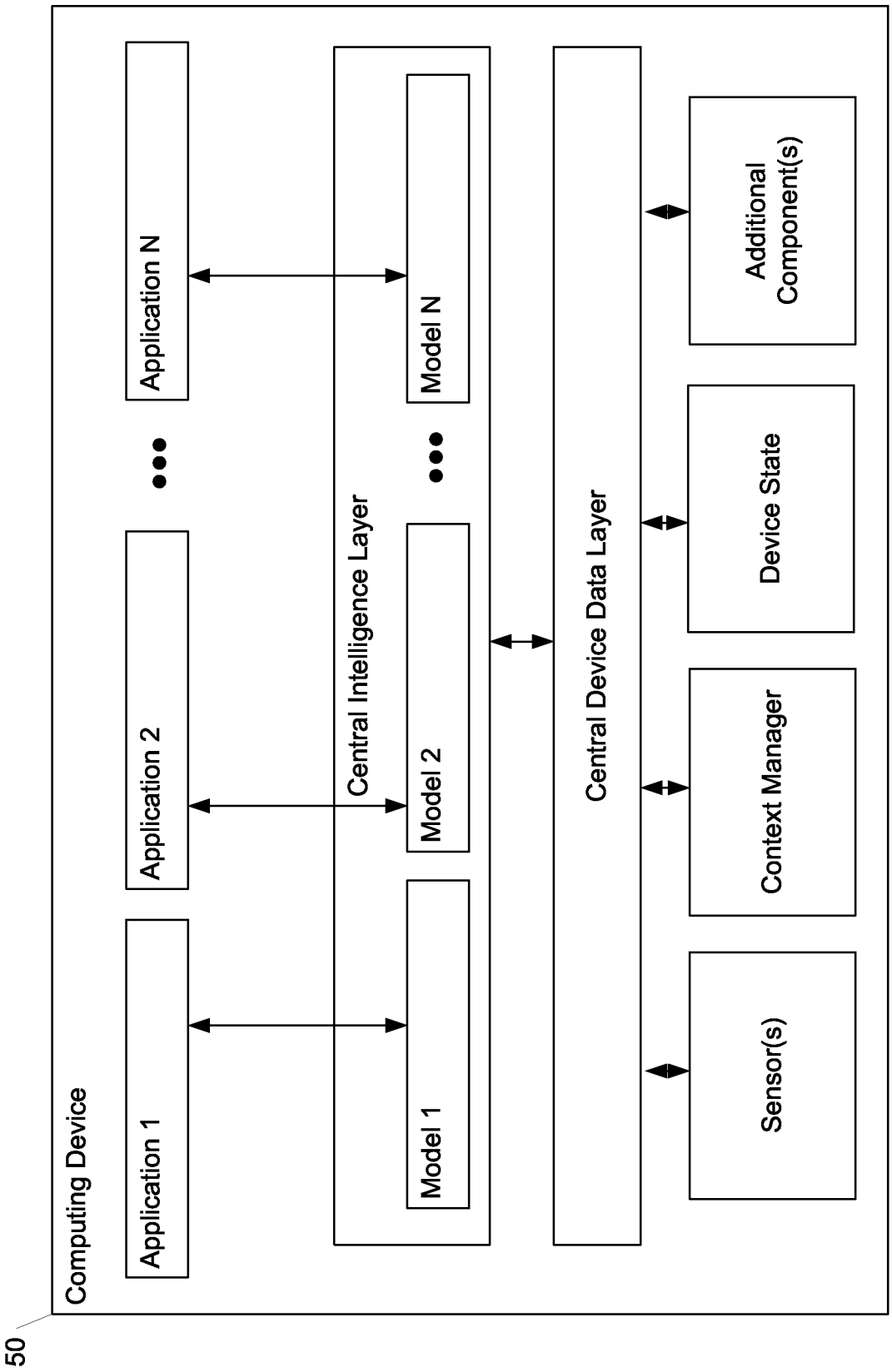

50
Computing Device
Application 1
Application 2
Application N
Central Intelligence Layer
Model 1
Model 2
Model N
Central Device Data Layer
Sensor(s)
Context Manager
Device State
Additional Component(s)

Figure 3B

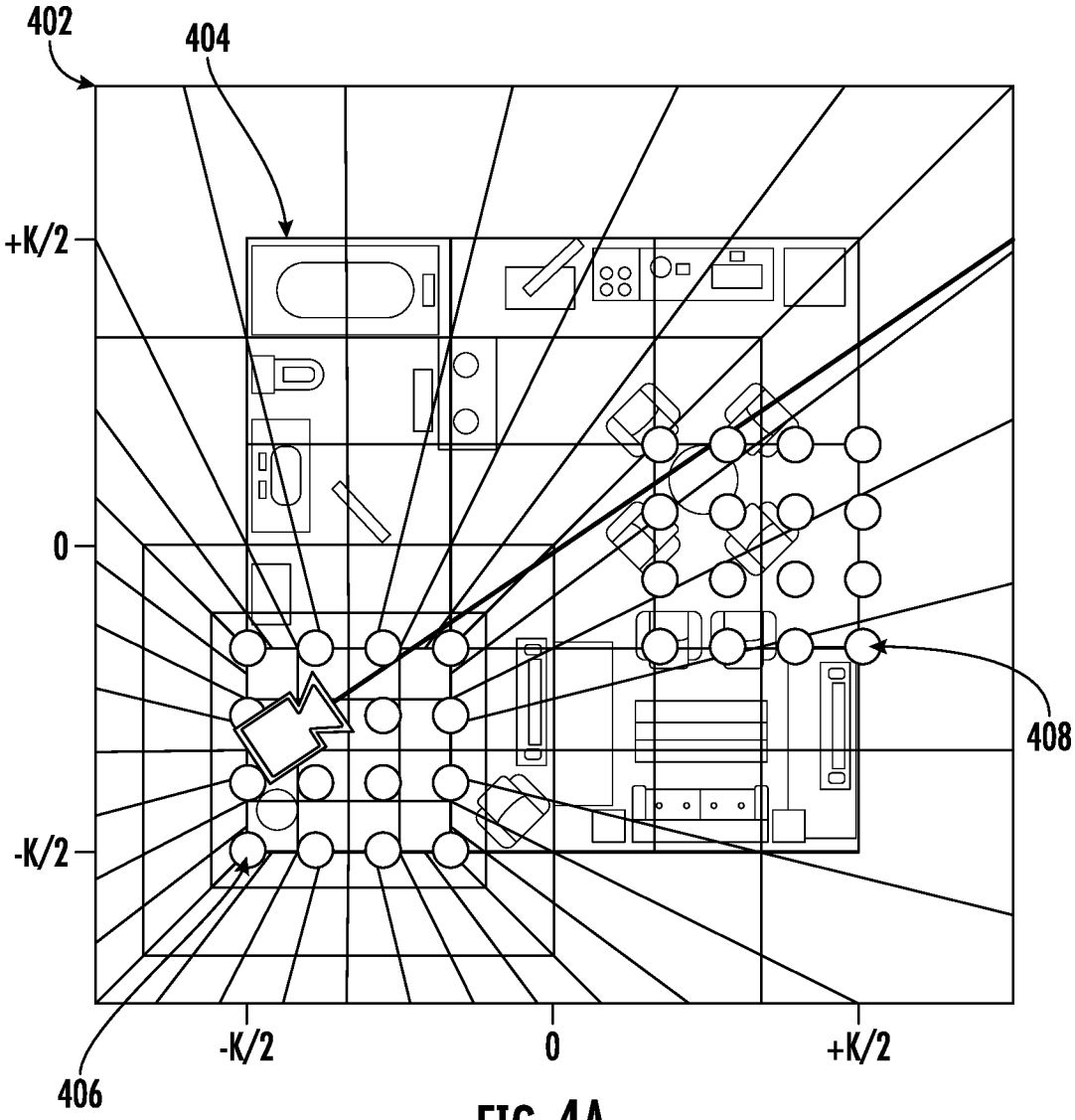


FIG. 4A

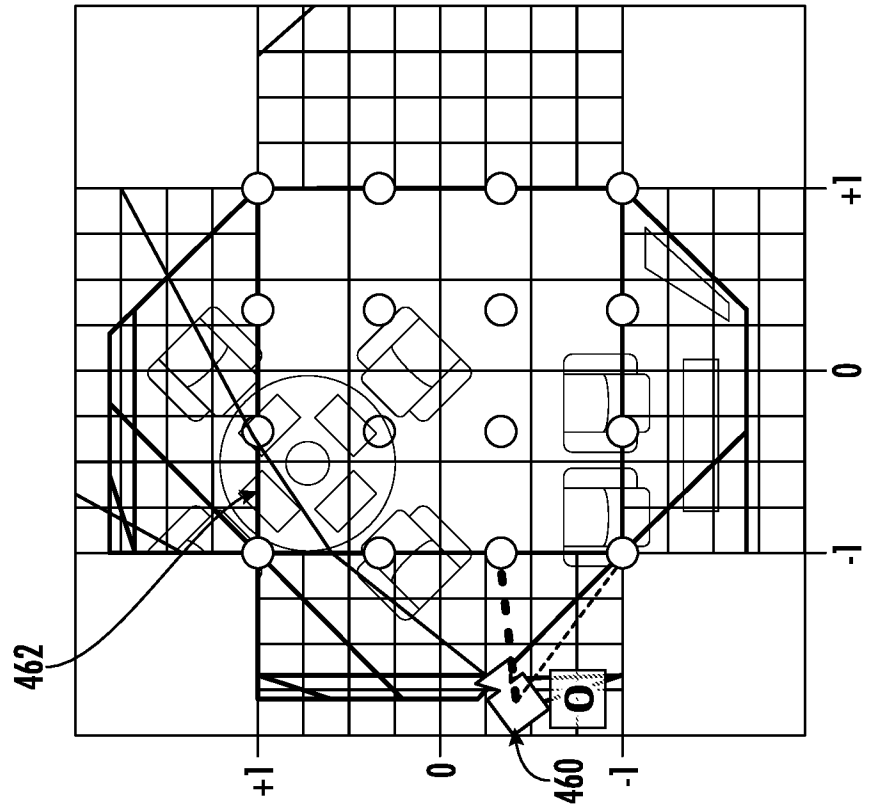


FIG. 4C

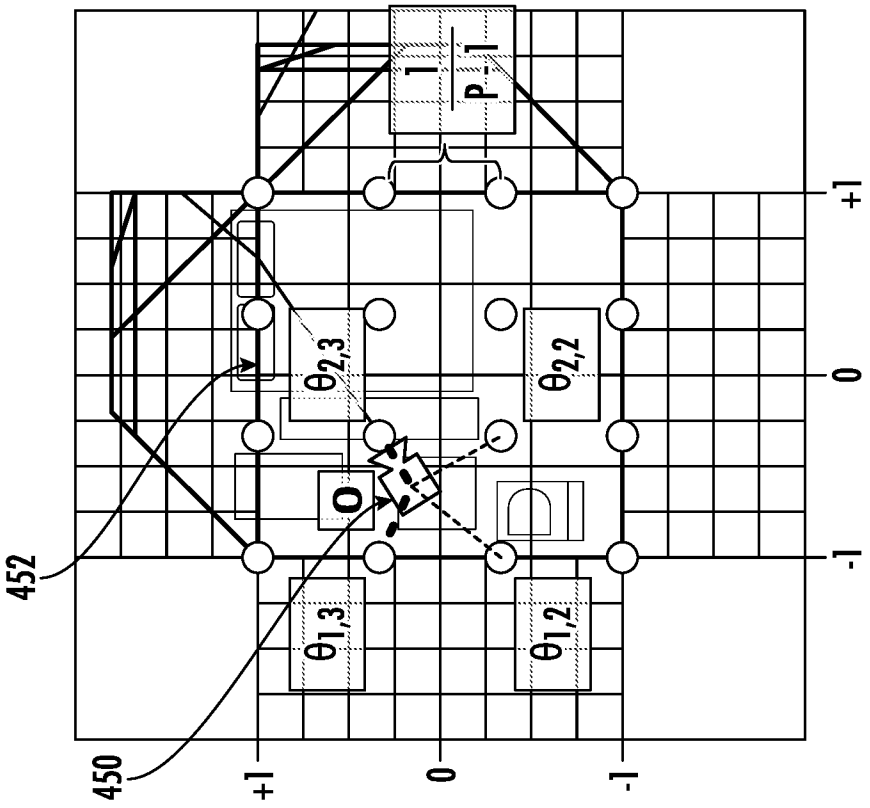


FIG. 4B

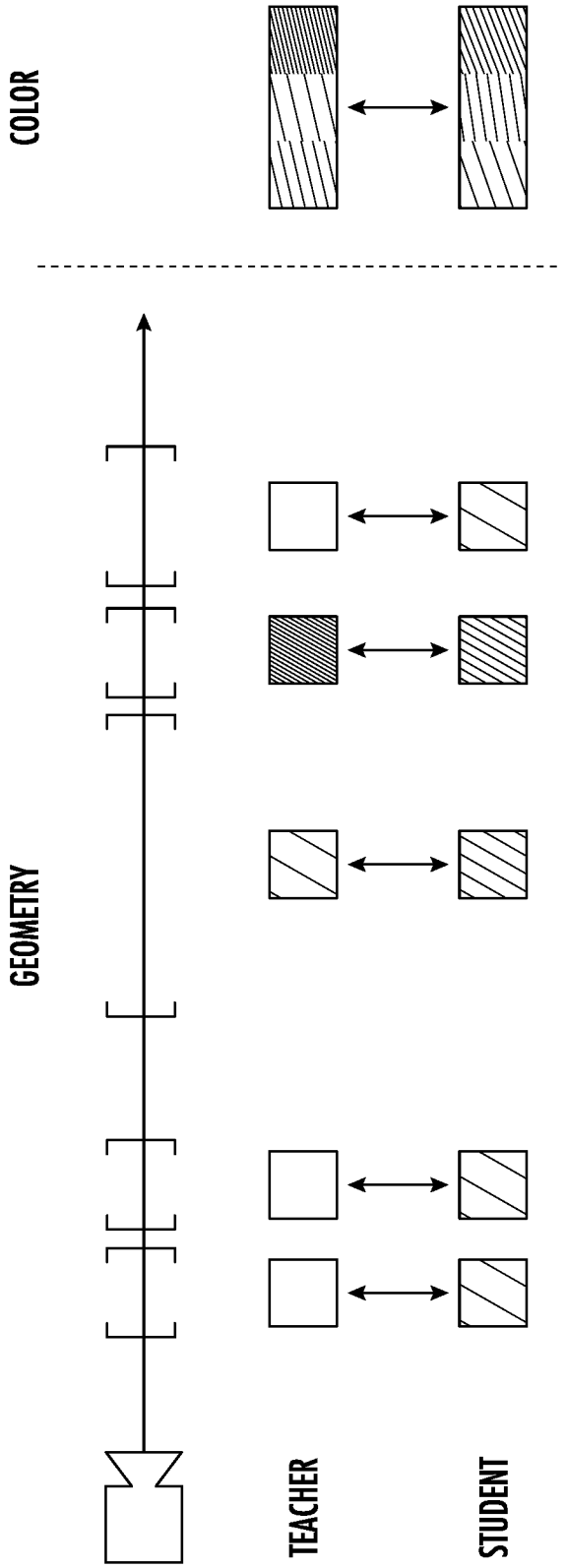


FIG. 5

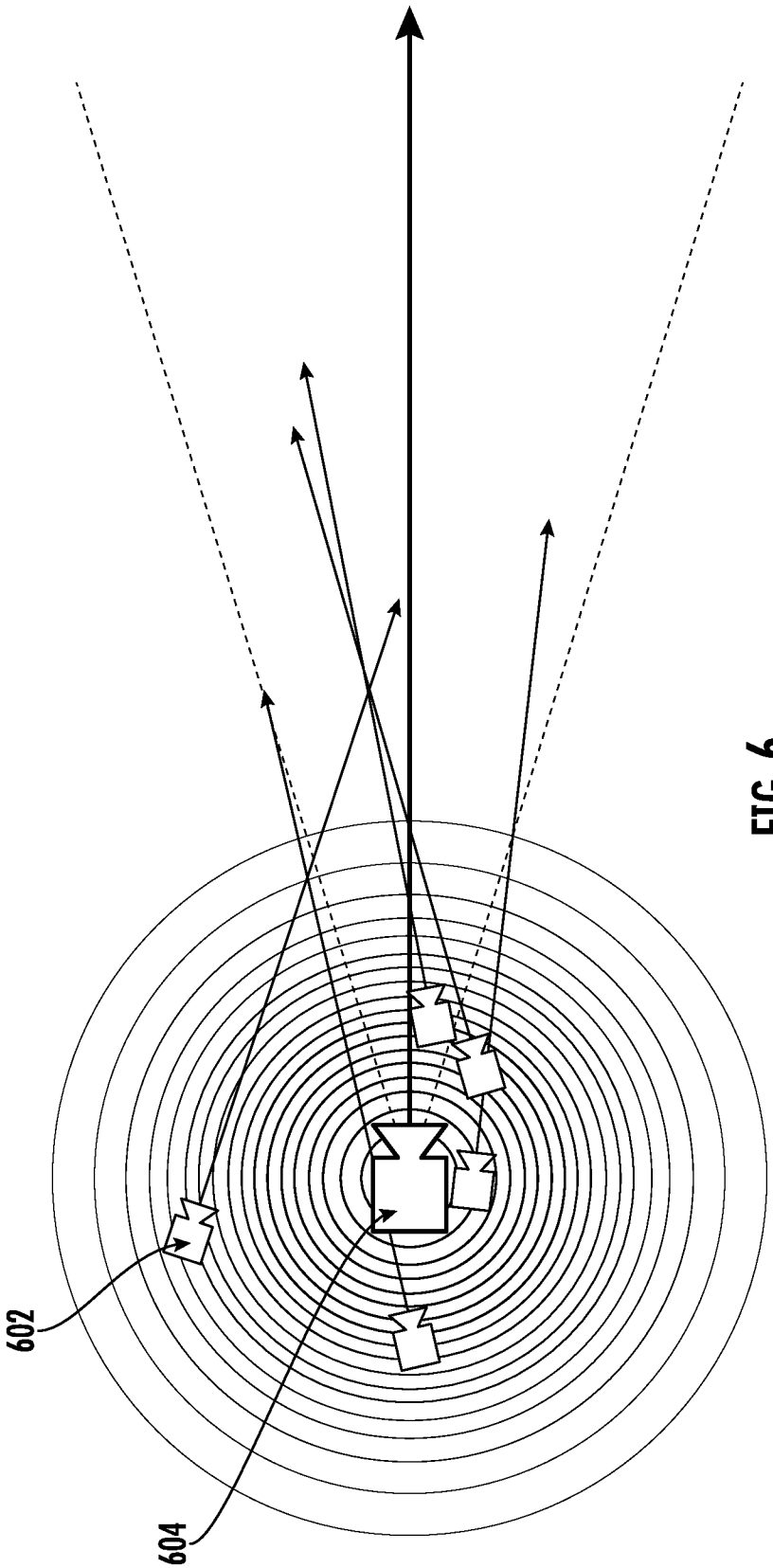


FIG. 6

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2024/060162

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06N3/045 G06N3/084 G06T15/00
ADD.

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06N G06T

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	REISER CHRISTIAN ET AL: "KiloNeRF: Speeding up Neural Radiance Fields with Thousands of Tiny MLPs", 2021 IEEE/CVF INTERNATIONAL CONFERENCE ON COMPUTER VISION (ICCV), IEEE, 10 October 2021 (2021-10-10), pages 14315-14325, XP034092578, DOI: 10.1109/ICCV48922.2021.01407 [retrieved on 2022-02-10]	1,8
Y	paragraph [03.1] - paragraph [03.3]	2-7

-/--

[x] Further documents are listed in the continuation of Box C. [ ] See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance;; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance;; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
14 March 2025	25/03/2025

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Falco, Gabriele

1

Form PCT/ISA/210 (second sheet) (April 2005)

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2024/060162

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	DUCKWORTH DANIEL ET AL: "SMERF: Streamable Memory Efficient Radiance Fields for Real-Time Large-Scene Exploration", ACM TRANSACTIONS ON GRAPHICS, vol. 43, no. 4, 12 December 2023 (2023-12-12), pages 1-13, XP093258007, US ISSN: 0730-0301	2-7
A	paragraph [05.1] - paragraph [05.3]; figures 2,3	1,8

1