PAVE

Samuel Leventhal*
samlev@cs.utah.edu
University of Utah School of Computing: Scientific Computing and Imaging Institute

Mark Kim
kimmb@ornl.gov
Oak Ridge National Lab

Dave Pugmire
pugmire@ornl.gov
Oak Ridge National Lab

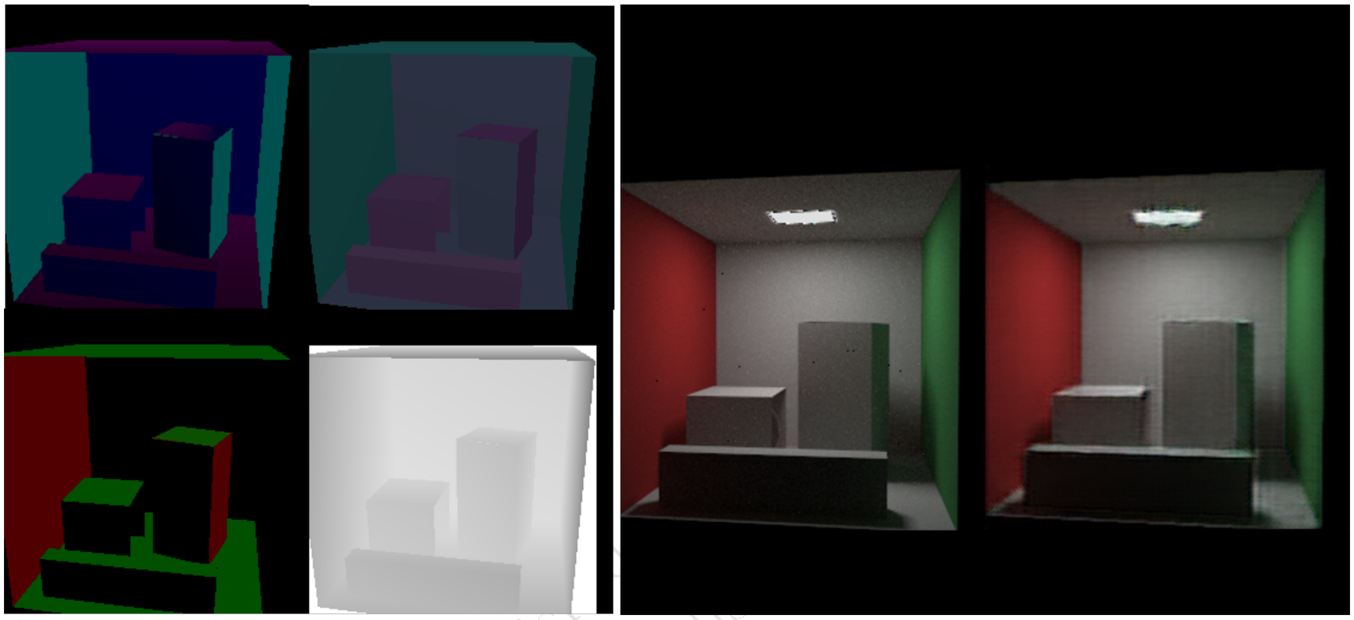


Figure 1: Rendered Conditional Geometry Buffers (**left set**) and artificial rendering with conditional generative adversarial neural network (**right couple**) comparing ground truth path traced rendering (**left**) with image generated (**right**).

ABSTRACT

The need for researchers and practitioners to have access to *in situ* operations is increasing for artificial intelligence and visualisations tasks as implementations become more complex and scaled widening the gap between computation time and IO throughput. What is more, a growing number of applications continue to be developed dealing with the combination of these two domains. With PAVE we present a solution which allows in place data management between visualisation and machine learning tasks. We then demonstrate our framework with the application of a conditional Generative Adversarial neural Network (cGAN) capable of in-situ and at scale training over path traced images resulting in a generative model able to produce real-time scene renderings with accurate light transport and global illumination of a quality comparable to offline approaches.

CCS CONCEPTS

• **Theory of computation → Parallel computing models; Distributed computing models; Structured prediction; Adversarial learning; Data structures and algorithms for data management**; *Probabilistic computation*; *Database query languages (principles)*; • **Applied computing → Computer-aided design**.

KEYWORDS

VTKm, neural networks, generative adversarial network, Adios, PyTorch, path tracing

ACM Reference Format:
Samuel Leventhal, Mark Kim, and Dave Pugmire. 2019. PAVE. In *Proceedings of ACM Conference (Conference'17)*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

As IO bandwidth continues to be significantly less than compute bandwidth in-situ methods have become essential. We hope to address the inverse relationship of IO and compute time by focussing on the intersection of two domains in which there is an increasing need for in-situ solutions in both applied and research communities, namely, searching scientific visualisation and machine learning applications. The purpose of this project is then to offer a solution which provides the means necessary to design machine learning implementations at scale and in-situ to aid or be the basis of visualisation and simulation tasks.

In this work we demonstrate and example application of PAVE's platform by employing a neural network for real time rendering and accurate light transport simulation within the framework of Python, specifically PyTorch, made compatible for distributed systems and high performance computing (HPC). The provided model is a coalescence of VTK-m, a visualisation toolkit fit for massively threaded architectures, PyTorch, an increasingly popular language within machine learning due to robust libraries for neural networks, and Adios2, an adaptable unified IO framework for data management at scale. The resulting work accomplishes this combination by utilising VTK-m to construct a path trace rendering tool able to fluidly and efficiently communicate to a conditional Generative Adversarial Network (cGAN) by means of Adios2 during training. The resulting generative model serves as a real-time filter for rendering images and visual simulations accurately approximating indirect illumination and soft shadows with quality comparable to offline approaches.

2 RELATED WORK

Real time true to life quality renderings of light transport remains an active area of research with a number of various approaches. To preserve real-time rates, previous works have stored precomputed radiance transfers for light transport as spherical functions within a fixed scene geometry which are then adjusted for varied light and camera perspective through projections within a basis of spherical harmonics [11]. Similarly, Light Propagation Volumes have been used to iteratively propagate light between consecutive grid positions to emulate single-bounce indirect illumination [6]. More recently, deep neural networks have been employed as a learned look up table for real-time rates with offline quality. With the use of convolutional neural networks Deep Shading is able to translate screen space buffers to into desired screen space effects such as indirect light, depth or motion blur. Similar to the methodology implemented in this work, Deep Illumination uses a conditional adversarial network (cGAN) to train a generative network with screen space buffers allowing for a trained network able to produce accurate global illumination with real-time rates at offline quality through a "one network for one scene" setting [12].

3 TECHNIQUE OVERVIEW

Utilisation of PAVE consists of three consecutive phases: rendering phase of conditional training images, training phase of the generative neural network, and execution phase of the trained network. Three core components, VTK-m, PyTorch, and Adios2 each fulfil a unique functional requirement during the separate stages. In this section we describe the independent design and global role each system plays.

3.1 System Overview

To achieve our goal of a conditional generative neural network capable of rendering geometric dependent object path simulations we begin by rendering informative conditional image buffers along with ground truth scene renderings. For this purpose the VTK-m was chosen due to its scalability and robust capability for HPC visualisation tasks. Provided the training set of conditional and ground truth images two neural networks, one convolutional and one generative, play a zero-sum game common to training GANs. To segue data management of training images the path tracer saves the training set in a distributed setting with the use of Adios2. During training PyTorch is then able to retrieve needed image data through the use of the adaptable IO provided by Adios's Python high-level APIs.

3.2 Path Tracer Design

Conditional image attributes and high quality ground truth rendered images are required for the training stage. For this reason the first stage of PAVE consists of generating a visual scene or simulation with VTK-m. Within the framework of VTK-m the implemented ray tracer renders images through means common to commercial ray tracers such as Monte Carlo sampling through probability distribution functions over shapes of interest and light intensity and pixel values with cumulative distribution function sampling, light scattering, randomly directed light paths and material sampling. The image buffers needed to compute light paths afford an informative conditional dependency on the behavior of lighting based on the geometry and light sources within a scene. These conditional buffers, namely albedo, direct lighting, normals of surfaces and depth with respect to point of view are then stored or passed to PyTorch with Adios2 APIs for C++ allowing for a a pipeline which preserves the solutions scalability.

3.3 Neural Network Design

The cGAN used closely follows that introduced by Thomas and Forbes with Deep Illumination [12]. Both the discriminator and generator network are deep convolutional neural networks implemented in PyTorch using training data retrieved from Adios files formatted and stored by the VTK-m path tracer. The training stage relies on four conditional buffers depth, albedo, normals and direct lighting along with an associated ground truth image of high light sample count and ray depth. Given the four conditional buffers the generator attempts to construct the ground truth image from noise. The discriminator is then fed both the generated and ground truth image. The loss used for the gradient back propagation update of both networks is based on the quality of the discriminators ability to classify the artificial and true image in which the generator is greater penalised when the discriminator accurately differentiates the two images, and similarly, the discriminator has a larger loss when incorrectly identifying real from fabricated images. The generator is then considered to have converged when the discriminator predicts both generated and true images with equal probability. For both discriminator and generator networks the activation functions used between layers is LeakyReLu and Sigmoid for the final layer [7]. Batch normalisation is also performed between internal layers to minimise covariant shift of weight updates and improve learning for the deeper networks used [3].

3.3.1 Generator Network. The generative network used is a deep convolutional network consisting of an encoder and decoder with skip connections-concatenations of equal depth layers within the encoding and decoding stages. Due to the illustrative 'shape' of this design the network is denoted a U-Net as introduced by Ronneberger et. al. for medical segmentation [10]. The motivation for utilising a U-Net is due to success of the skip connections in capturing geometric and spatial attributes by linking the decoded convolutional process to the encoded upconvolutional. The mapping of contracting feature segmentation onto expanding upsampling within the network allows us to also exploit nearness to object geometry within the constructed and target image through Euclidean distance. As a result, performance in terms of required training time, quality of preserved structure, and accuracy maintaining light information of generated images drastically improves with the addition of an L1 loss to the classic binary cross entropy common to training adversarial networks when updating the discriminator and generator [5][2].

3.3.2 Discriminator Network. For discriminating between artificial and ground truth image renderings a deep convolutional patchGAN network is used motivated by the added advantage of providing a patch-wise probability of an image in question as being real or fake. The benefit of a patch-wise probability allows for higher regional accuracy within an image as well as applicable for image-to-image tasks as introduced by Isola et. al. [4]. The image classification probability is then interpreted as the average of these patch wise probabilities over an image in question.

As input during training the discriminator network is given the set of conditional space buffers along with either the visualisation generated by the cGAN generator network or the ground truth global illumination rendering produced with the VTK-m path tracer. Taking into account the conditional image buffers stacked atop an image sample an image sample under question the resulting input is a tensor of the form Width x Height x 15. The discriminator then attempts to predict with what probability the provided image stack, either fabricated by the U-Net or the VTK-m path tracer, is real.

Based on the discriminators performance the loss is computed using the classic loss for training GANs along with an L1 loss. Training is complete, and the generator has converged, when the discriminator predicts images as real or fake with an equal probability, e.g. 50% chance of accuracy. At this point the discriminator network is discarded and the resulting generator affords a real-time in situ visualisation tool able to produce accurate global illumination from conditional geometric buffers.

3.4 Core Design Pattern

PAVE offers a means of in-situ communication between visualisations or simulations requiring an artificial intelligence component. The resulting solution consists of two core components, simulation and learning. Pave then allows each task to communicate results among each other fluidly. The scientific simulation or visualisation task developed by the user can then provide resulting visualisations to a separately developed learning model as input or employ a learning model within the visualisation task.

3.4.1 User Provided Simulation or Visualisation. For our in-situ and scalable solution the scientific simulation task designed by the user can produce results to be passed to a learning model or save the simulation data to be retrieved later by the learning model. With IO managed by Adios2 the user simulation can remain fully scalable to distributed systems. For this same reason in the provided example in section 3.5 we chose VTK-m arrays as the data structures of the visualisation task.

```
class PAVE
{
public:
    using type = vtkm::Float32;
    using visualisation = vtkm::cont::ArrayHandle<type>
    void open(std::string &file);
    void save(visualisation &v);
    void visualize(vtkm::rendering::Canvas &canvas);
};
```

The code example demonstrates the visualisation component in our solution. The user is able to open or generate visualisation data to be saved or sent to other processes.

3.4.2 User Defined Machine Learning Application. PAVE allows researchers or practitioners to implement their learning algorithms in the increasingly popular language Python due to having a robust library for learning tasks and notably neural networks.

```
import PAVE
from torch.utils.data import DataLoader

def load_data(visualisation_variable=None, path = None):
    if path:
        train_set  = PAVE.AdiosDataLoader(path)
        train_data = DataLoader(dataset = train_set)
        return train_data
    else:
        PAVE.open(visualisation_variables)

def train(Model, simulation_variables):
    for (i, var) in enumerate(visualisation_variables):
        train_sample = load_data(var)
        prediction = Model(train_sample)
        ...
```

Above we demonstrate employing PAVE while training a PyTorch model. The training method for Model is able to request visualisation samples used during training based on some parameter used in the visualisation task or retrieve precomputed visualisation data by calling PAVE.

3.4.3 Communication of Visualisation Data and Learning. As demonstrated in section 3.4.1 PAVE allows for the user to save or pass data produced by the simulation and similarly the user would also be able to request results from the learning model depending on the application. Section 3.4.2

3.5 Our Provided Example of PAVE

For our PyTorch in situ proposal the current systems we employ are VTK-m, for rendering path traced images and conditional geometry buffers focus, and Adios2 for data management. We discuss the design pattern for our in situ visualisation task with support from deep learning in the order of operations followed within the pipeline of use. Namely, we present the design pattern for rendering light transport in VTK-m coupled with data transport to PyTorch (3.5.1). Subsequently we explain the infrastructure for embedding VTK-m throughput managed by Adios2 within PyTorch and demonstrate through example (3.5.2) by instantiating a PyTorch data interface allowing for data parallelization and multi-GPU/distributed training as used within the framework for training our cGAN model.

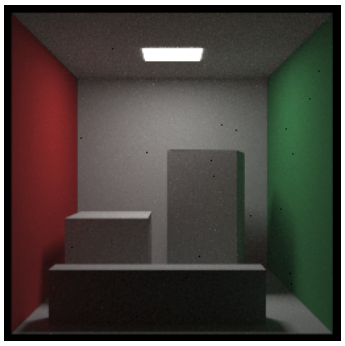
3.5.1 VTK-m Light Transport Visualisation. Path traced images are maintained within C++11 as VTK-m arrays which can be passed by reference directly to PyTorch using Adios2 APIs or written to Adios2 .bp file and retrieved during training or when utilising the neural network to generate novel scene renderings from rendered geometry buffers.

3.5.2 PyTorch cGAN Global Illumination Generator. For training, our solution used by the cGAN is "*AdiosDataLoader*", a data class inheriting from the abstract indexing class PyTorch *torch.utils.data.Dataset*. The *AdiosDataLoader* employs Adios2 to either retrieve from file or have passed by reference vector representations of path traced images and conditional buffers. Within VTK-m during generation these vectors represented as VTK-m vectors and within PyTorch as numpy arrays. In this manner the training or test sets needed by PyTorch and created by VTK-m are available to PyTorch in situ or with reference to written memory. If retrieving VTK-m's renderings PyTorch will compile Adios2 attributes from file as tabled by Adios2 into .bp files. VTK-m generated datasets can be retrieved with *read_adios_bp()* or passed to a similar *get_adios_bp()* and subsequently forwarded to our *get_split()* to partition the dataset into 60% training, 20% testing and 20% validation subsets. The split datasets are then used to construct the *AdiosDataLoader* class which inherits from the *torch.utils.data.DataLoader* thereby providing a data sampler of our VTK-m renderings with a single-process or multi-process iterator over the dataset affording the tools necessary to train our neural networks in the canonical manner.

In the above code sample the *AdiosDataLoader* class is used to partition the data set into training, validation and testing as well as offer a distributable data sampler with an array-like data structure allowing index access to elements and collection size functionality.

4 CORNELL BOX EXPERIMENT

To evaluate the quality of in situ deep learning aided visualisations train the cGAN networks on rendered images of a Cornell box, a commonly used 3D modelling framework for quality assessment. We train the model using renderings of the Cornell box with high light sample count and depth computation per ray for various camera angle perspectives into the box along with the associated image geometry buffers for a given camera orientation. We then assess the quality of the models final generated renderings looking at the accuracy of global illumination. We then also demonstrate the performance of the models ability to render global illumination when given image buffers for a novel scene not used for training similar in content but not exact. The scene used for training is comprised of the classic set up with one overhead light source in the center of a white ceiling, a white back wall and a white floor. The remaining walls are then colored red on the left and green on the right in order to afford different colored light transport and demonstrate diffuse interreflection. The contents of the Cornell box are three cuboids of various shapes and sizes to provide diverse shading and diffused lighting.



The conditional differed shading geometry buffers used are direct lighting, normal planes, depth and albedo as shown in figure 1. The geometry buffers serve as joint variables for the conditional probability distribution which the global illumination path traced images are considered to exist. The conditional arguments in this experiment then aid the cGAN in learning behavior of light paths given the geometry of a scene in question.

5 RESULTS

5.1 Cornell Box Experiment Results

The resulting generated images show promising results for deep learning aided in situ scientific visualisation. We observe the network successfully learned to emulate light transport in a realistic fashion with offline performance **EXAMPLE**. What is more, though designed in a “one network for one scene” setting, the generative net proved to be adaptive and able to generate accurate renderings for not only unobserved camera orientation renderings during training but also varied scenes of a similar flavor when provided the conditional geometry buffers of the novel scene.

5.2 Solution Design Assessment

To render ___ 256x256 images with VTK-m on 2 ____ GPUs required ___ hours. Training the cGAN on this image data set over ___ epochs on the same machine took ___ hours. Once trained the run time of applying the generative U-Net provided the conditional buffer set averages ___ seconds.

6 CONCLUSIONS

Our work offers a distributable and scalable implementation and framework to allow researchers and practitioners to easily integrate state of the art deep learning tools afforded by the approachable PyTorch and robust visualisation resource VTK-m for in situ graphics rendering or scientific simulation for HPC systems. With the presented example application utilising cGANs for generative visualisation also offers the prospect of quickly and accurately visualise conditionally dependent data such as light path global illumination’s dependence on scene geometry. Subsequently we here then also offer a framework for visualisation within C++ with the use of VTK-m as well as Python in tandem or independently.

ACKNOWLEDGMENTS

This research was supported in part by an appointment to the Oak Ridge National Laboratory ASTRO Program, sponsored by the U.S. Department of Energy and administered by the Oak Ridge Institute for Science and Education.

REFERENCES

[1] Ken Dahm and Alexander Keller. 2017. Learning light transport the reinforced way. *arXiv preprint arXiv:1701.07403* (2017).

[2] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. Advances in Neural Information Processing Systems. (2014).

[3] Sergey Ioffe and Christian Szegedy. 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* (2015).

[4] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. 2017. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition.* 1125–1134.

[5] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. 2017. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition.* 1125–1134.

[6] Anton Kaplanyan and Carsten Dachsbacher. 2010. Cascaded light propagation volumes for real-time indirect illumination. In *Proceedings of the 2010 ACM SIGGRAPH symposium on Interactive 3D Graphics and Games.* ACM, 99–107.

[7] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. 2013. Rectifier nonlinearities improve neural network acoustic models. In *Proc. icml*, Vol. 30. 3.

[8] Mehdi Mirza and Simon Osindero. 2014. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784* (2014).

[9] Kenneth Moreland, Christopher Sewell, William Usher, Li-ta Lo, Jeremy Meredith, David Pugmire, James Kress, Hendrik Schroots, Kwan-Liu Ma, Hank Childs, et al. 2016. Vtk-m: Accelerating the visualization toolkit for massively threaded architectures. *IEEE computer graphics and applications* 36, 3 (2016), 48–58.

[10] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention.* Springer, 234–241.

[11] Peter-Pike Sloan, Jan Kautz, and John Snyder. 2002. Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. In *ACM Transactions on Graphics (TOG)*, Vol. 21. ACM, 527–536.

[12] Manu Mathew Thomas and Angus G Forbes. 2017. Deep Illumination: Approximating Dynamic Global Illumination with Generative Adversarial Network. *arXiv preprint arXiv:1710.09834* (2017).