Introducing TensorFlow Graphics: Computer Graphics Meets Deep Learning

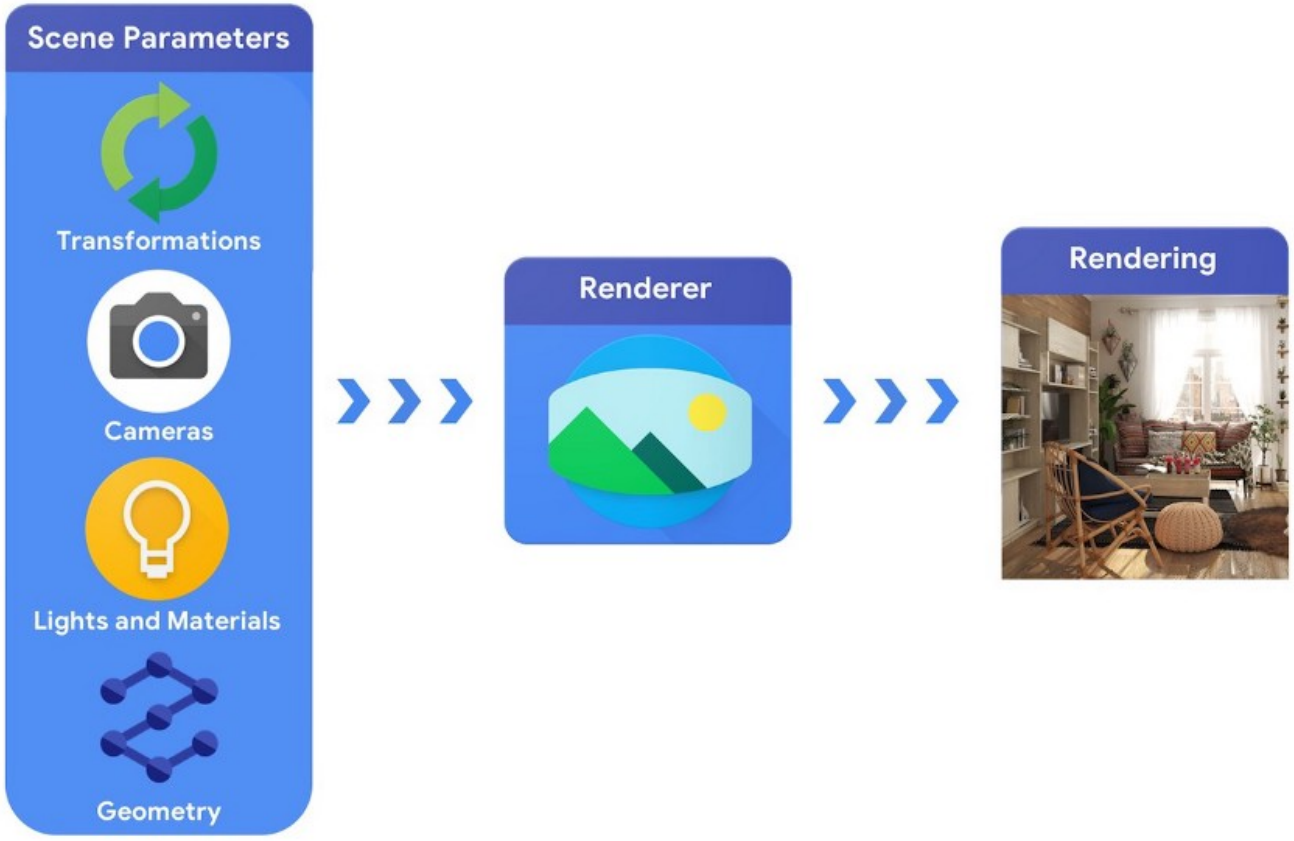
TensorFlow Follow
May 10, 2019 · 5 min read

Posted by Julien Valentin and Sofien Bouaziz

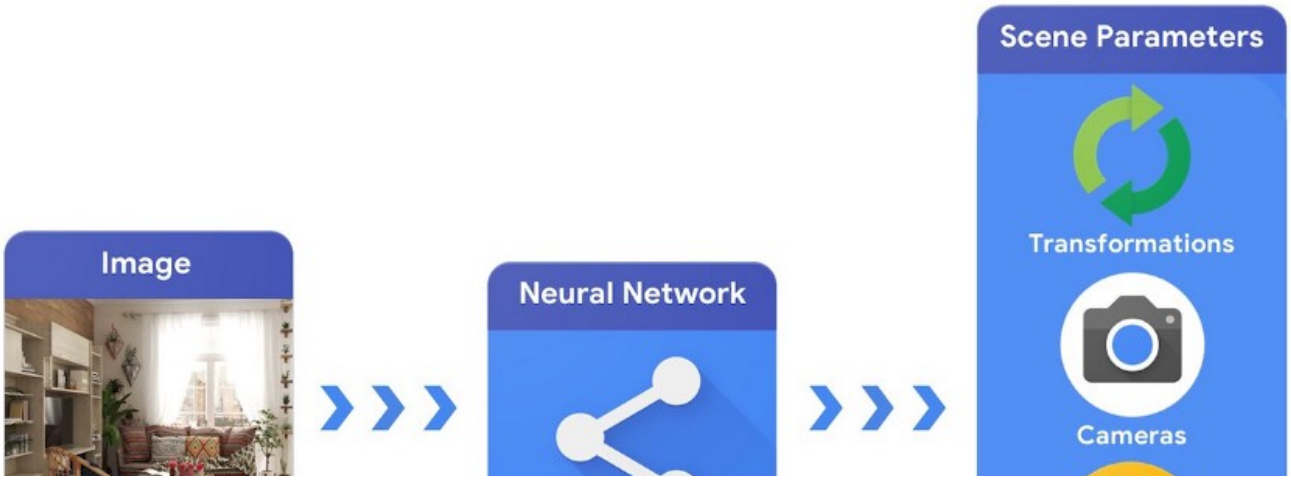
Github repository: https://github.com/tensorflow/graphics

The last few years have seen a rise in novel differentiable graphics layers which can be inserted in neural network architectures. From spatial transformers to differentiable graphics renderers, these new layers leverage the knowledge acquired over years of computer vision and graphics research to build new and more efficient network architectures. Explicitly modeling geometric priors and constraints into neural networks opens up the door to architectures that can be trained robustly, efficiently, and more importantly, in a self-supervised fashion.

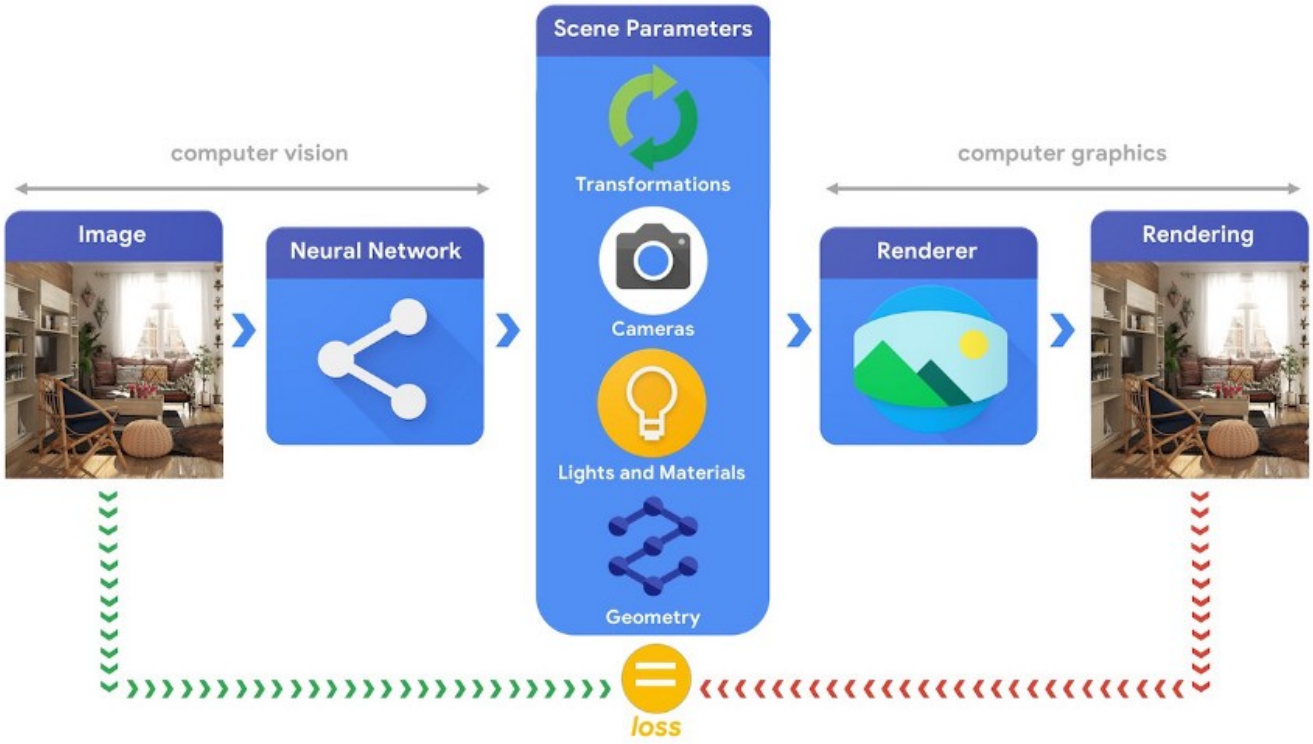
At a high level, a computer graphics pipeline requires 3D objects and their absolute positioning in the scene, a description of the material they are made of, lights and a camera. This scene description is then interpreted by a renderer to generate a synthetic rendering.



In comparison, a computer vision system would start from an image and try to infer the parameters of the scene. This allows the prediction of which objects are in the scene, what materials they are made of, and their three dimensional position and orientation.



Training machine learning systems capable of solving these complex 3D vision tasks most often requires large quantities of data. As labelling data is a costly and complex process, it is important to have mechanisms to design machine learning models that can comprehend the three dimensional world while being trained without much supervision. Combining computer vision and computer graphics techniques provides a unique opportunity to leverage the vast amounts of readily available unlabelled data. As illustrated in the image below, this can, for instance, be achieved using analysis by synthesis where the vision system extracts the scene parameters and the graphics system renders back an image based on them. If the rendering matches the original image, the vision system has accurately extracted the scene parameters. In this setup, computer vision and computer graphics go hand-in-hand, forming a single machine learning system similar to an autoencoder, which can be trained in a self-supervised manner.

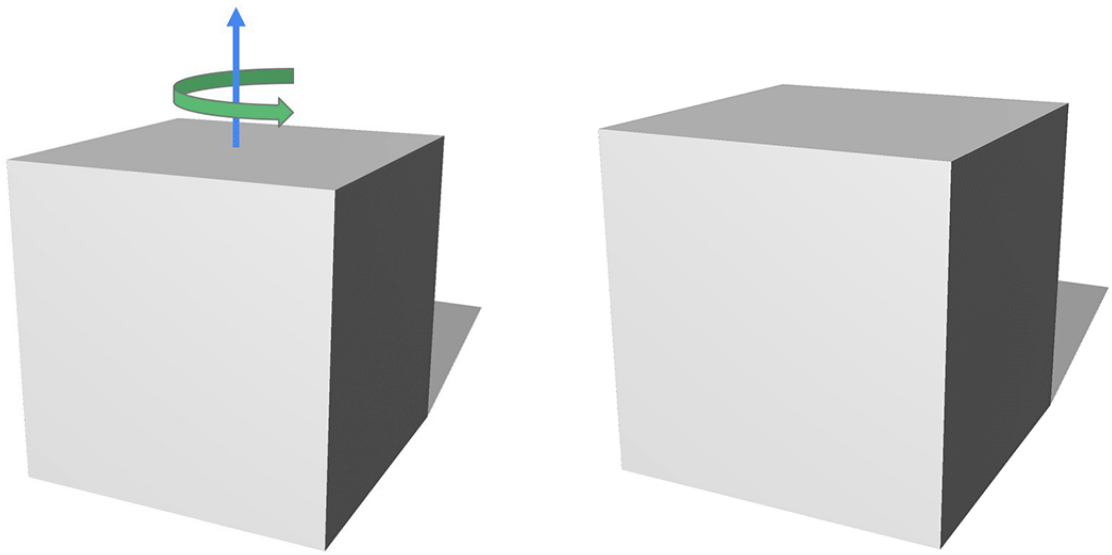


Differentiable Graphics Layers

In the following, we will explore some of the functionalities available in TensorFlow Graphics. This tour is not exhaustive; for more information visit our Github to discover the new possibilities made available by TensorFlow Graphics.

Transformations

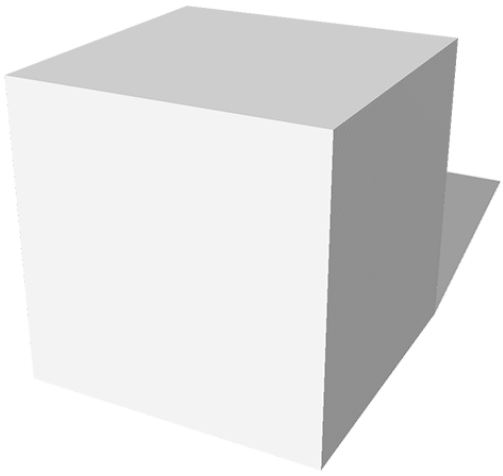
Object transformations control the position of objects in space. In the illustration below, the axis-angle formalism is used to rotate a cube. The rotation axis is pointing up and the angle is positive, leading the cube to rotate counterclockwise. In this Colab example, we show how rotation formalisms can be trained in a neural network that is trained to both predict the rotation and translation of an observed object. This task is at the core of many applications, including robots that focus on interacting with their environment. In these scenarios, grasping objects (e.g. by their handle) with a robotic arm requires a precise estimation of the position of these objects with respect to the arm.



Modelling cameras

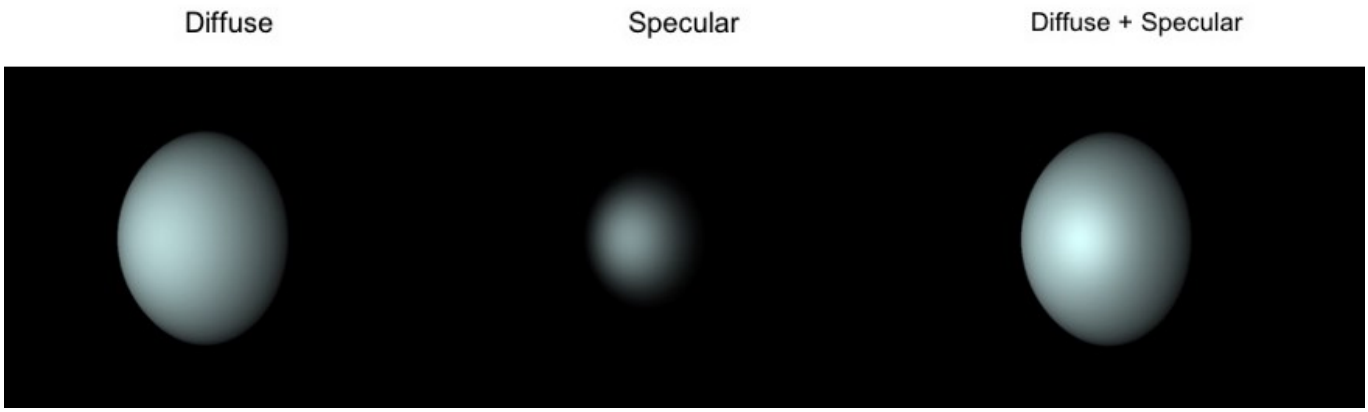
Camera models play a fundamental role in computer vision as they greatly influence the appearance of three dimensional objects projected onto the image plane. As can be observed below, the cube appears to be scaling up and down, while in reality the

changes are only due to changes in focal length. Try this Colab example for more details about camera models and a concrete example of how to use them in TensorFlow.



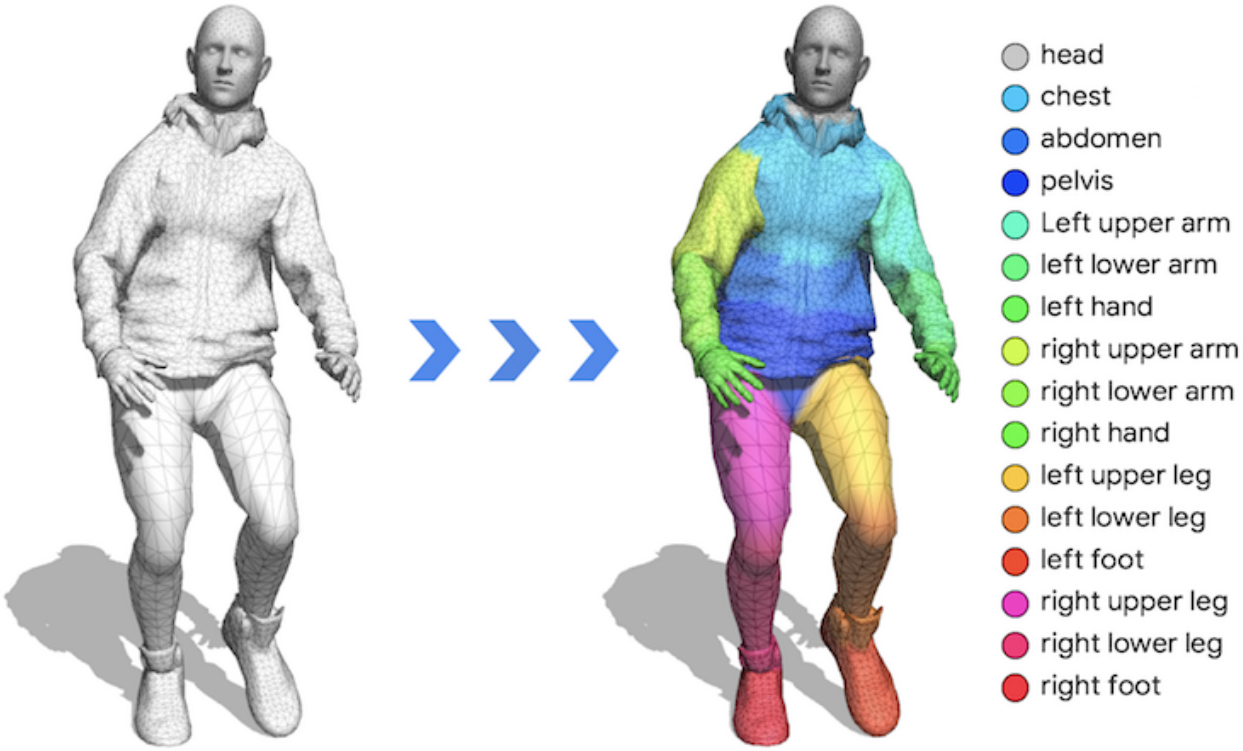
Materials

Material models define how light interacts with objects to give them their unique appearance. For instance, some materials like plaster reflect light uniformly in all directions, where others like mirrors are purely specular. In this interactive Colab notebook, you will learn how to generate the following renderings using Tensorflow Graphics. You will also have the opportunity to play with the parameters of the material and the light to develop a good sense of how they interact. Accurately predicting material properties is fundamental to many tasks. For instance, it can allow users to drop virtual furniture in their environment and have the pieces photo-realistically blend with their interior, giving users an accurate perception of how that piece of furniture would look.



Geometry — 3D convolutions and pooling

In recent years, sensors outputting three dimensional data in the form of point clouds or meshes are becoming part of our everyday life, from smartphone depth sensors to self driving car lidars. Due to their irregular structure, convolutions on these representations are significantly harder to implement compared to images which offer a regular grid structure. TensorFlow Graphics comes with two 3D convolution layers, and one 3D pooling layer, allowing for instance the training of networks to perform semantic part classification on meshes as illustrated below and demonstrated in this Colab notebook.



TensorBoard 3d

Visual debugging is a great way to assess whether an experiment is going in the right direction. To this end, TensorFlow Graphics comes with a TensorBoard plugin to interactively visualize 3d meshes and point clouds.

Get started

The first release of TensorFlow Graphics supports is compatible with TensorFlow 1.13.1 and above. You will find the API and instructions to install the library by visiting https://www.tensorflow.org/graphics.

Acknowledgments

Creating TensorFlow Graphics was a team effort. Special thanks to Cem Keskin, Pavel Pidlypenskyi, Ameesh Makadia, and Avneesh Sud who all made significant contributions.

Machine Learning 3d Graphics Unsupervised Learning Announcements