

Monocular Depth-Estimation for Autonomous Driving using Neural Networks

Proposed by: Bardh Rushiti
Mentored by: Dr. Thomas Kinsman

Checkpoint

Monocular depth estimation is a challenging problem in computer vision that aims to predict depth information from a single image. In my project, I first tried to find working code for monocular depth estimation using C++ and OpenCV. After some research, I found a promising repository on GitHub called "monodepth2" developed by Niantic Labs. This repository contained pre-trained models and most of the active code-base was in Python and Jupyter Notebooks.

However, I encountered some challenges when trying to run the code on my machine using C++ and OpenCV. I discovered that there are some intermediary steps that enable us to convert models to something that is runnable by OpenCV. This intermediary file is known as the ".onnx" file, which essentially contains information about the network, and training weights, and is able to make inferences in C++/OpenCV. The result can be seen in the image down below:



*Images are concatenated in a video for depth estimation

After successfully running inference, I realized that the process was too slow (less than 1fps) since it was running on CPU. Therefore, the next step in this process was to enable the model to run on GPU by setting up the OpenCV-DNN module with CUDA backend support in Windows. This optimization improved the inference speed significantly.


The code for this existing inference can be found in this submission as 'inference.cpp' and 'models\model-small.onnx' (). This is a C++ file that uses the OpenCV library and the DNN module in it to run the .onnx file (pre-trained neural network model). The program defines a string variable file_path that specifies the path to the directory containing the pre-trained neural network files. It then sets a string variable model to the name of the neural network file to be used for depth estimation. It then enters a loop

that reads in each frame of the video, applies the necessary resize and scaling to enable the neural network to make meaningful predictions, and then applies the neural network to estimate the depth map of the scene.

In the meantime, I have also been reading the relevant papers and trying to understand how the model is generating maps for relative depth and how the neural network is trained. Specifically, I am studying the papers "Towards Robust Monocular Depth Estimation: Mixing Datasets for Zero-shot Cross-dataset Transfer" and "Digging Into Self-Supervised Monocular Depth Estimation". These papers provide valuable insights into the training process and the neural network architecture used in the "monodepth2" repository.

Finally, my ultimate goal is to retrain a model from the NYU-v2 and KITTI datasets. These datasets contain thousands of annotated images with corresponding depth maps, and they are commonly used for monocular depth estimation tasks. To achieve this, I plan to use transfer learning to fine-tune the pre-trained models provided in the "monodepth2" repository. This will involve adapting the model architecture, training the model on the new datasets, and evaluating the performance of the new model. By doing so, I hope to improve the accuracy and robustness of the monocular depth estimation model for real-world applications.

References:

1. [GitHub - nianticlabs/monodepth2: \[ICCV 2019\] Monocular depth estimation from a single image](https://github.com/nianticlabs/monodepth2)
2. Enabling model run on GPU by setting up CUDA and cuDNN for my machine.
 -  [Setup OpenCV-DNN module with CUDA backend support on Windows](#)
 - [Setup OpenCV-DNN module with CUDA backend support \(For Windows\) | by Techzizou | Geek Culture | Medium](#)
 - [\(optional\) Exporting a Model from PyTorch to ONNX and Running it using ONNX Runtime](#)