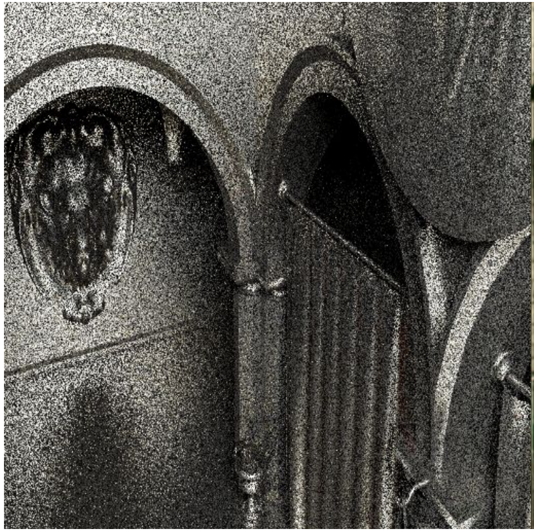
How we achieved “Real” time denoising over the weekend

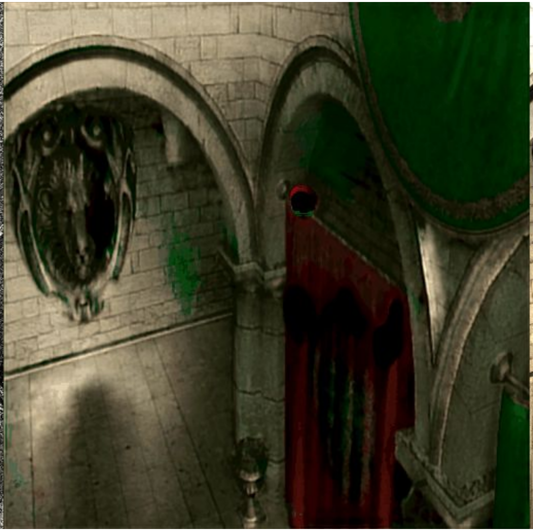
CIS 565 Milestone 3 - Deep Regret
Vaibhav Arcot & Dewang Sultania

https://etsin.fairdata.fi/dataset/0ab24b68-4658-4259-9f1d-3150be898c63/data - Dataset Link

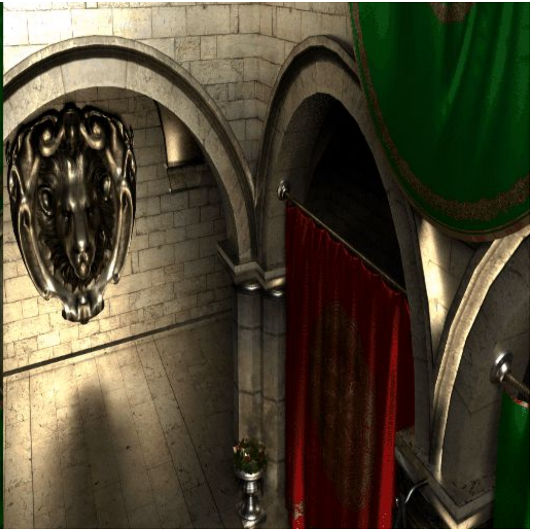
Out of the box results



1 spp



Denoised



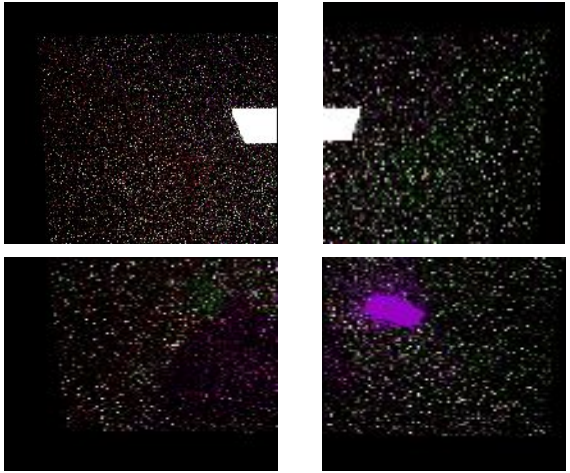
2500 spp

Ingredients for “Real” time denoising

- Tiling
- Torchscript
- C++ Torch frontend
- Glm vectors, torch tensors, cv matrices
- Efficient use of memory
- TensorRT (Final presentation)

Tiling

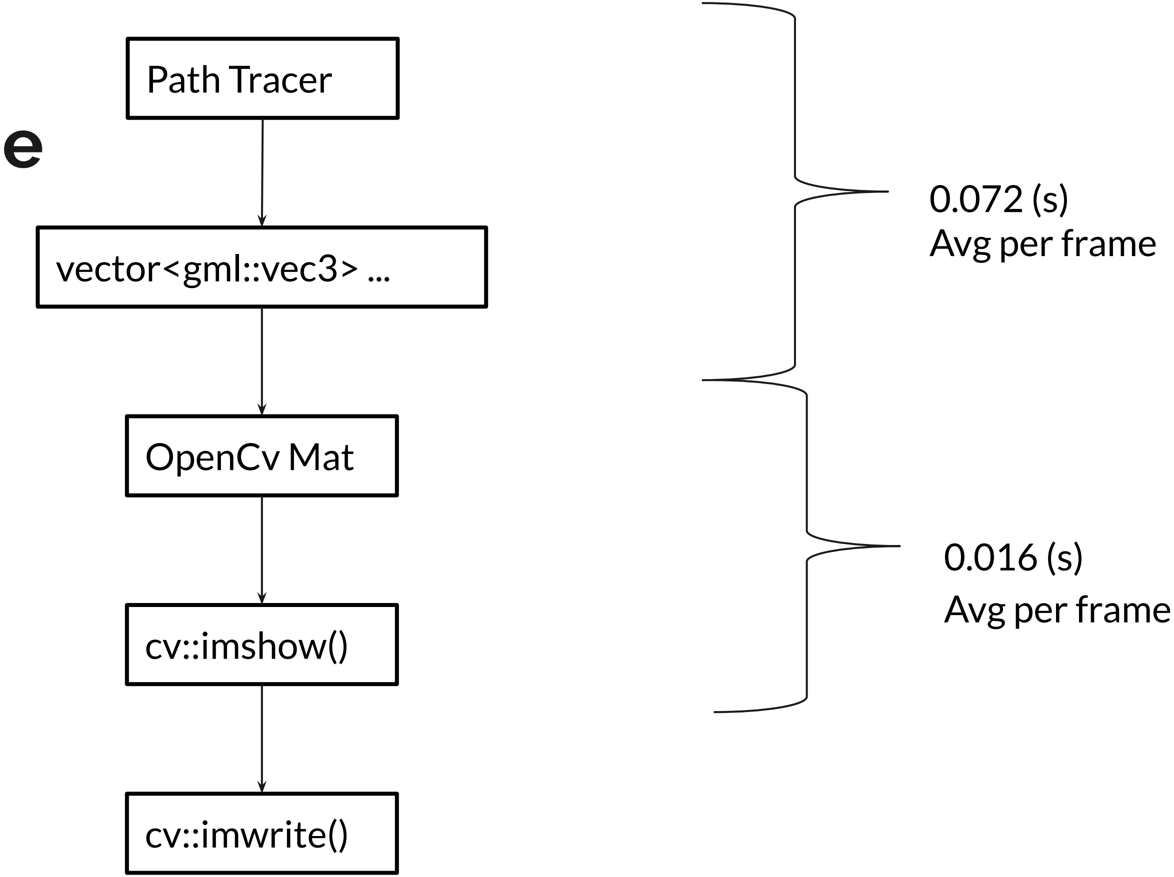
- We got full res images to work!!
- How?
  - Tiling - We took random crops of size 256*256 from the image and sent that to the network, so each input image is basically a sequence of 4 smaller images.



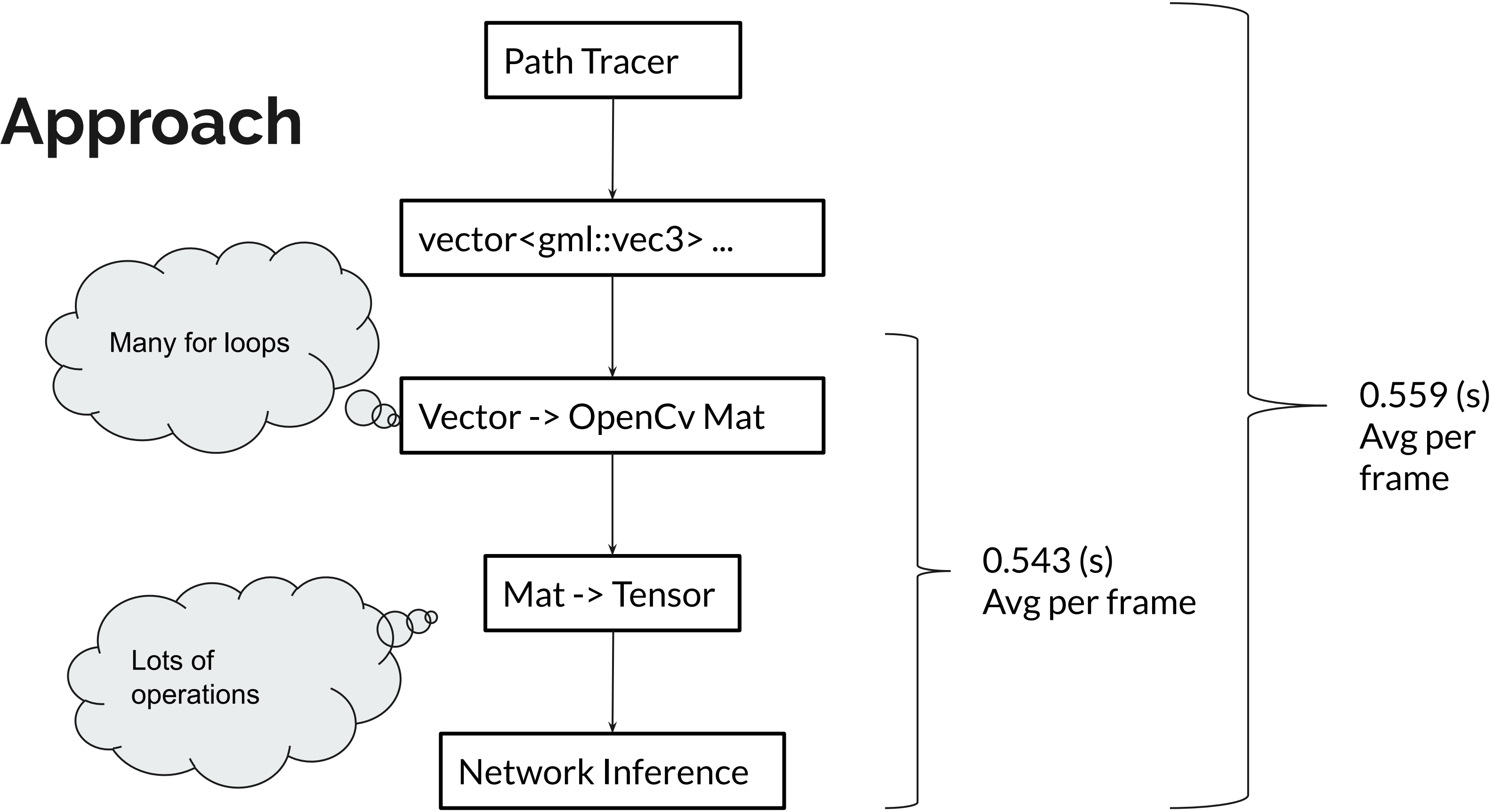
TorchScript

- Converts Python models to a format that can be read without python
- Incrementally transitions a model from a pure Python program to a TorchScript program that can be run independently from Python, such as in a standalone C++ program.
- Makes it possible to train models in PyTorch using familiar tools in Python and then export the model via TorchScript to a production environment where Python programs may be disadvantageous, for performance and multi-threading reasons.

Data Gen Pipeline



First Approach



Path Tracer

vector<gml::vec3> ...

Vector -> OpenCv Mat

Mat -> Tensor

Network Inference

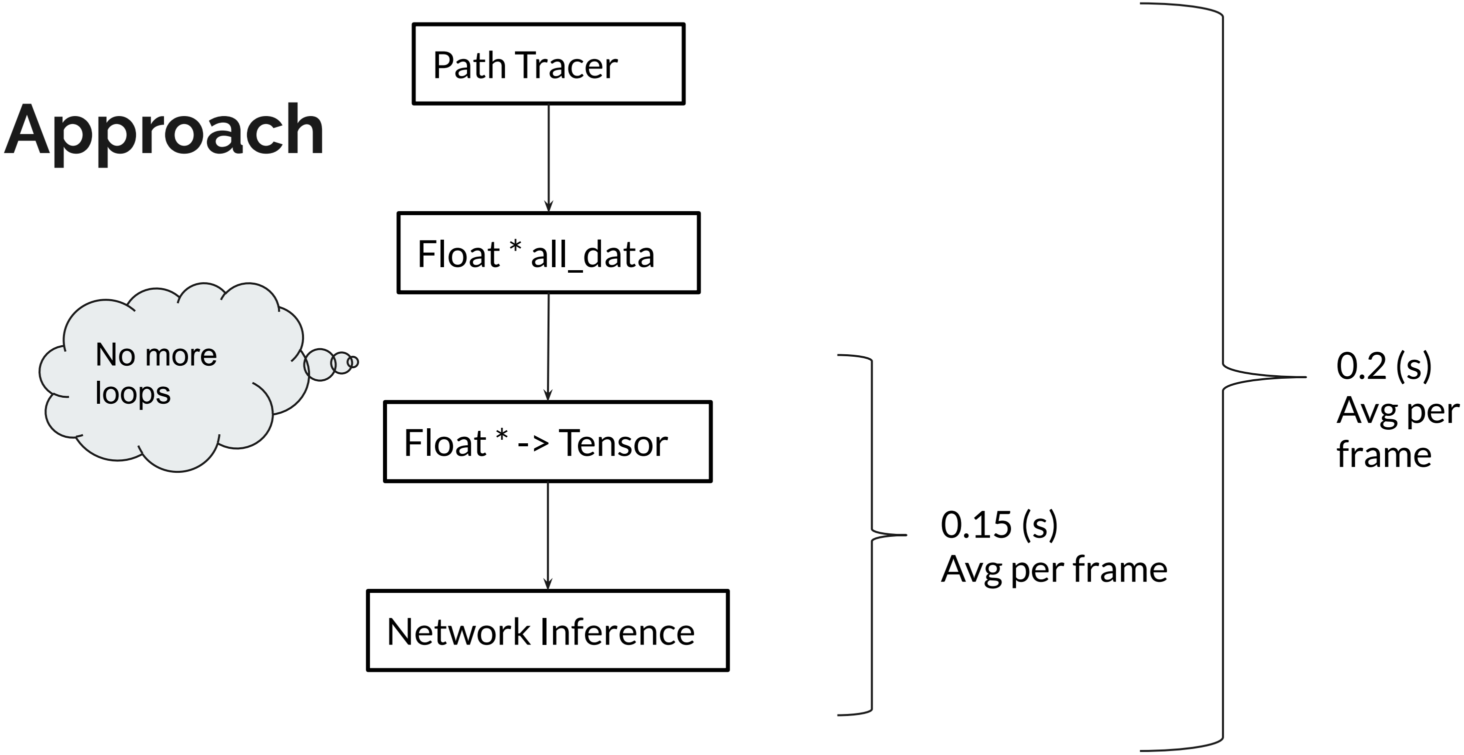
Many for loops

Lots of
operations

0.543 (s)
Avg per frame

0.559 (s)
Avg per
frame

Second Approach



Path Tracer

Float * all_data

Float * -> Tensor

Network Inference

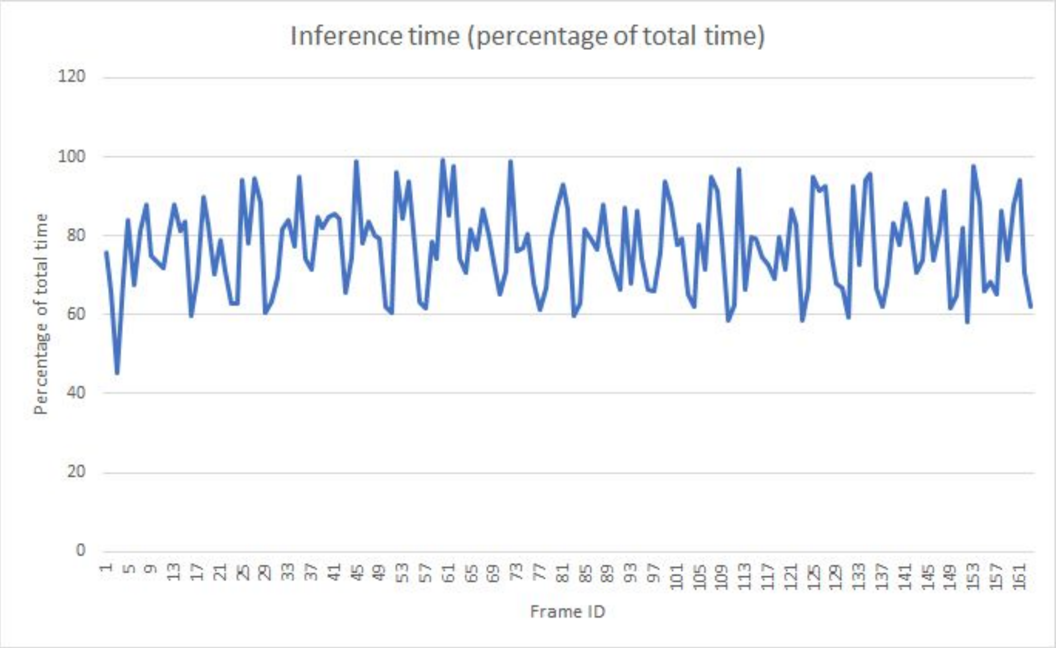
No more loops

0.15 (s)
Avg per frame

0.2 (s)
Avg per
frame

Live Demo

Case for TensorRT



Future Work

- Integrate TensorRT and redo the benchmarks for the realtime setup
- Finish rendering the other demo scenes
- Train more