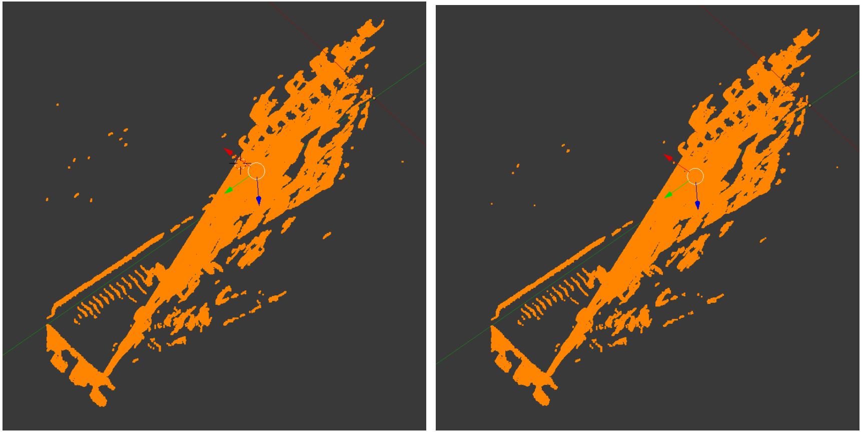
Processing of 3D Scans using Machine Learning

Lukáš Gajdošech

21.07.2020 - 01.08.2020

Retrained Results



(a) original (b) processed

Benchmark

GPU	API	backend	train time	inference time
Vega64	OpenCL	Plaid-ML	246.9s	–
Vega64	OpenCL	TF (ROCm)	30.8s	70ms
RTX 2060S	CUDA	TensorFlow	31.44s	36ms
RTX 2080	CUDA	TensorFlow	?	?

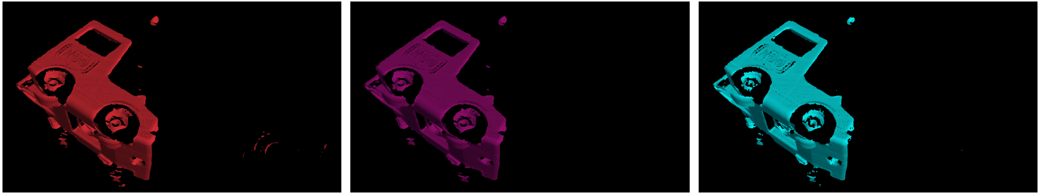
Masks

(a) original

(b) processed

(c) processed

Visual Evaluation



(a) original (b) processed (c) processed

Inspiration

- Havaei, M., Davy, A., Warde-Farley, D., Biard, A., Courville, A., Bengio, Y., Pal, C., Jodoin, P.-M., and Larochelle, H.
  Brain tumor segmentation with deep neural networks.
  Medical Image Analysis 35 (Jan 2017), 18–31.
- Iglovikov, V., and Shvets, A.
  Ternausnet: U-net with vgg11 encoder pre-trained on imagenet for image segmentation, 2018.
- Long, J., Shelhamer, E., and Darrell, T.
  Fully convolutional networks for semantic segmentation, 2014.
- Noh, H., Hong, S., and Han, B.
  Learning deconvolution network for semantic segmentation, 2015.
- Ronneberger, O., Fischer, P., and Brox, T.
  U-Net: convolutional networks for biomedical image segmentation.
  vol. 9351, pp. 234–241.

Overview

for $t \in [1, T]$ and performance of the learned model does not meet requirement **do**

Local Gradient Computation:

(a) Data holder P_k, $\forall k \in [1, K]$, generates pseudo-identity P_k^*

(b) P_k calculates its local gradient:

- P_k retrieves current $\mathbf{w}_t$ from Block B_t
- P_k calculates local gradient using Eq. (6)
- P_k applies a differential privacy scheme to the local gradient using Eq. (10)
- P_k normalizes the gradient using Eq. (11), Eq. (12)

(c) P_k broadcasts the message msg_k:

$$\mathbb{M}_{P_k}^t = (P_k^*, \nabla \hat{g}_k(\mathbf{w}_t)^*, s_k^t, t)$$

(a) original

Global Gradients Aggregation:

(a) Computing node C_j competes to solve PoW problem

(b) If it wins, C_j updates the predictive model:

- C_j retrieves local gradients from its memory pool
- C_j finds the sum gradient descent direction calculated by Eq. (14)
- C_j selects the l-nearest local gradients based on their cosine distances calculated by Eq. (15)
- C_j calculates the global gradient using Eq. (16)
- C_j updates the model using Eq. (8)

(c) C_j creates a new block B_{t+1} containing the following data:

$$B_{t+1} = (\mathbf{w}_{t+1}, [\mathbb{M}_{P_1}, \mathbb{M}_{P_2}, \ldots, \mathbb{M}_{P_K}])$$

a block contains hash, nounce, etc. What we show here is only the data load.

end for

(b) processed

PU-Net: Point Cloud Upsampling Network

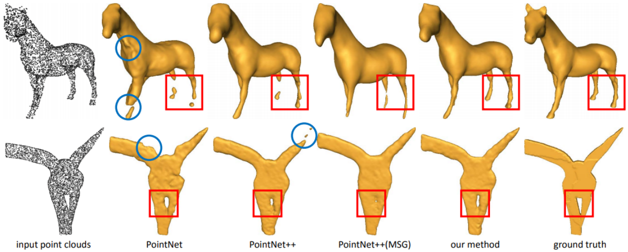


Figure 5. Surface reconstruction results from the upsampled point clouds.

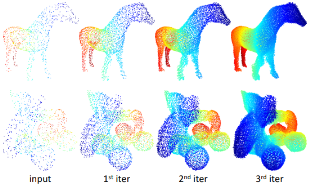


Figure 6. Results of iterative upsampling. We color-code points by the depth information. Blue points are closer to us.

pare the reconstruction results of different methods with the direct Poisson surface reconstruction method [16] provided in MeshLab [5]; see Fig. 5. We can observe that the reconstruction result from our method is the closest to the ground truth, while other methods either miss certain structures (*e.g.*, the leg of the Horse) or overfill the hole.

Results of iterative upsampling. To study the ability of our network to handle varying number of input points, we design an iterative upsampling experiment, which takes the output of the previous iteration as the input of the next iteration. Fig. 6 shows the results. The initial input point cloud has 1024 points and we increase fourfold in each iteration. From the results, we can see that our network can produce

Coming up

- study and run PU-Net
- find and implement (for Keras) appropriate evaluation metric (Intersection-Over-Union ?)
- filter and reconstruct whole dataset