DVGROb: Direct Voxel Grid Robust Optimization

May 2, 2024

Roberta Giorgi

Abstract

In this project I propose a Neural Network approach based on NeRF (Mildenhall et al., 2020) models, where the aim is to reconstruct tumor spheroids structure starting from the dusty photos of them taken at different angles.

1. Introduction

Tumor spheroids are 3D tumor cells aggregates used in cancer treatment research. Their three-dimensionality allows to better investigate tumors characteristics and behaviours, thus enabling superior clinical and pharmaceutical analysis to find a cure (Hamilton, 1998). However, inspecting their structure and their volume, an extremely important information (Goldmacher & Conklin, 2012), is not easy. The current instruments allow to take pictures of the spheroids as they fall and rotate, but they are prone to dirt, as dust, being trapped between the camera and the spheroid, resulting in a spotted photo. To this end, possible candidates are Neural Networks and in particular NeRF-like models, modified in order to be more robust to these inevitable artifacts.

Code. The Python code for this project can be found at the following GitHub repository: `https://github.com/rolorgi/DVGROb.git`.

2. Related work

NeRF NeRFs (Mildenhall et al., 2020) are the latest Neural Networks for novel view synthesis, using MLPs to reconstruct the scene as a volume density of the object starting from various pictures of it taken at different angles. The scene is constructed using a ray casted through it, from which N points are sampled and RGB values predicted for each point. The pictures form a kind of sphere surrounding the scene, such that the ground truths of the RGB values are the picture pixel colors that are hit by the ray. They are trained using the classical MSE loss between the rendered and true RGB values of N points on the ray.

DVGO DVGO (Sun et al., 2022) is a modified NeRF able to train way faster than the original. This improvement is achieved by implementing a voxel density reconstruction before the original MLP, thus avoiding to train the empty spaces outside the voxels.

RobustNeRF The problem with classic-purpose NeRF models is the a priori assumption that the dataset (i.e. the images) used is flawless, with no artifacts and disturbance. This problem has been addressed by (Sabour et al., 2023) in their RobustNeRF. The loss is modified with a mask that takes into consideration the nature of the outliers (i.e. the artifacts) during training. In fact, outliers (as dust) are spacial consistent transient objects. They result in non-independent pixels which location is not coherent inside the scene. By exploiting this information they have been able to develop a weights mask to filter out unwanted ones.

Non Neural Networks approaches I've found no other Neural Network approach for the spheroid reconstruction problem, although other solutions have been proposed. For example (Chen et al., 2014) developed a software application based on image/signal processing algorithms, using width and length to calculate spheroids volumes.

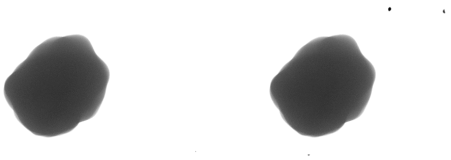
3. Material and Methods

Dataset and Blender The dataset used is a series of photos taken from an *in silico* reproduction of a spheroid as a Blender (Community, 2018) file. The photos with the needed infomation (camera angle and orientation) have been taken thanks to the BlenderNeRF add-on (Raafat, 2023). I've followed the format and number of data originally used in DVGO (and NeRF itself). In Figure 1 you can see two example of the same photo, with and without dirt.

Code The whole Network have been written in Python using the PyTorch framework (Paszke et al., 2019). Since modest resources were needed, I used the free version of Google Colaboratory with the T4 GPU to train the Network.

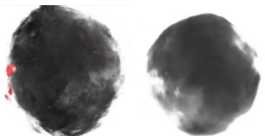
Email: Roberta Giorgi <giorgi.1846342@studenti.uniroma1.it>.

Deep Learning and Applied AI 2023, Sapienza University of Rome, 2nd semester a.y. 2022/2023.



(a) Clean spheroid (b) Dusty spheroid

Figure 1. Two dataset example. In 1a you can see the clean capture, while in 1b you can see the same capture, but with added dirt.



(a) DVGO trained (b) DVGROb trained

Figure 2. The same dust dataset trained with DVGO 2a and DVGROb 2b. In 2a you can see the red dust spotted, while in 2b you see the same timestamp and notice that the red dust is gone.

4. Results

DVGROb The proposed Neural Network is the original DVGO model modified to include the RobustNeRF loss, thus becoming Direct Voxel Grid Robust Optimization. Moreover, since the original aim was not to get the 3D model but only novel renderings, I implemented a fine reconstructed model export to extrapolate the resulting point cloud and being able to form its convex hull to calculate the volume using Open3D (Zhou et al., 2018). These modifications are coded in the `run2.py` and `vis_volume2.py` files, the former including the fine model export ad the modified loss, the latter to visualize the final 3D model and calculate the volume. The code for RobustNeRF was present in their GitHub page, but it was written in Jax (Bradbury et al., 2018) and implemented for a 2D loss. Since in DVGO the loss is calculated on the RGB values of the ray points, the loss object is a 1D N long vector of MSE values. So, following the formulas found in their paper and starting from the provided code, I implemented it in PyTorch and applied the formulas to the required 1D loss object, while keeping coherent results. The robust loss modification is implemented as a N long binary vector mask applied to the MSE loss, where the values 1 or 0 are the results of the various formulas applied to the loss. The original MSE was calculated using the proper PyTorch Neural Network functional, but in order to be able to apply the mask, I manually implemented it using PyTorch modules. Finally, I changed voxel density threshold hyperparameters in `default.py` to make it work.

Training The modified loss was successfully integrated and the Network applied to the dusty dataset to be compared to the clean one. The latter was not trained with DVGROb but only with the naive DVGO, since as (Sabour et al., 2023) state, when the robust loss is applied to a clean scene it removes non-artifacts information. I also trained the same dusty dataset but with a red-colored dust in order to be able to see the differences in the rendered video. In Figure 2 you can indeed see that the red dust has been captured inside the model and that DVGROb was able to remove it.

Finally, in Table 1 you can find the summarized DVGO results while Table 2 summarized DVGROb results.

Table 1. DVGO performance

Dataset	Clean	Dusty	Dusty Colored
Testing PSNR	19.08	19.39	19.77
Volume	1042	1196	1083

Table 2. DVGROb performance

Dataset	Dusty	Dusty Colored
Testing PSNR	19.45	20.12
Volume	623	577

5. Conclusion

DVGROb is able to remove the dust from the model, achieving comparable PSNR results. In Table 1 we can also notice that the dusty dataset lead to larger models, expecially in the dataset with normally grey-colored dust, suggesting that the dust has been wrongly included in it. However, the volume obtained is not optimal. First, I haven't been able to detect its unit of measure, probably being dependent on the input file one. Secondly, the volume is reduced way too much, suggesting us that the mask applied is too strict and removes too many weights. This conclusion can also be seen in Figure 2, where the robust optimization leads to a smoother surface and empty spots where they shouldn't exist. The rendering is not compact in DVGO either, but DVGROb increases this problem. I suppose that the spheroid is an object too self-similar such that reconstructing its finer details is an extremely complex task, made even worse when very specific yet again similar artifacts like normal dust are present. Given the results differences between colored and normal dust, I suppose that to get better results one can work on the dusty dataset beforehand instead of directly using the Neural Networks, for example using image segmentation algorithms to distinguish the dust and exploit this information during training.

References

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

Chen, W., Wong, C., Vosburgh, E., Levine, A. J., Foran, D. J., and Xu, E. Y. High-throughput image analysis of tumor spheroids: A user-friendly software application to measure the size of spheroids automatically and accurately. *Journal of Visualized Experiments*, (89), Jul 2014. doi: 10.3791/51639.

Community, B. O. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. URL http://www.blender.org.

Goldmacher, G. V. and Conklin, J. The use of tumour volumetrics to assess response to therapy in anticancer clinical trials. *British Journal of Clinical Pharmacology*, 73 (6):846–854, May 2012. ISSN 1365-2125. doi: 10.1111/j.1365-2125.2012.04179.x. URL http://dx.doi.org/10.1111/j.1365-2125.2012.04179.x.

Hamilton, G. Multicellular spheroids as an in vitro tumor model. *Cancer Letters*, 131(1):29–34, September 1998. ISSN 0304-3835. doi: 10.1016/s0304-3835(98)00198-0. URL http://dx.doi.org/10.1016/s0304-3835(98)00198-0.

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., and Ng, R. Nerf: Representing scenes as neural radiance fields for view synthesis, 2020.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library, 2019.

Raafat, M. BlenderNeRF, May 2023. URL https://github.com/maximeraafat/BlenderNeRF.

Sabour, S., Vora, S., Duckworth, D., Krasin, I., Fleet, D. J., and Tagliasacchi, A. Robustnerf: Ignoring distractors with robust losses, 2023.

Sun, C., Sun, M., and Chen, H.-T. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction, 2022.

Zhou, Q.-Y., Park, J., and Koltun, V. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018.