Research Project: Applying NerfMM to fNIRS

Amit Sandler and Tomer Porian

18th March 2022

Abstract

In this project, we researched novel machine learning algorithms and analyzed how they might be applied to the challenge of aligning an fNIRS helmet. fNIRS is a novel new alternative to fMRI, the most common tool used in diagnosing various diseases of the brain, especially those which involve blood The device is non-intrusive, producing high-resolution images using strong magnets. However, it does have a few drawbacks:

- The system is large and heavy
- It requires that the person being examined not move at all, which is difficult to ask of a baby or a toddler

In addition, fMRI cannot be used on patients with certain kinds of pacemakers and other implants in their bodies. Thus, fNIRS may be a better solution- instead of a large, bulky machine, it's comprised of a simple helmet that rests upon the patient's head. It also uses IR lights instead of magnets. fNIRS does have its own drawbacks, namely that the helmet has to sit perfectly on the patient's head in order for it to work. Solving this issue was the focus of our project, in which we attempted to track the helmet's exact location using NeRFMM[WWX+21], a method for generating novel views of complex scenes without known parameters. A solution of this sort would greatly improve the usefulness of fNIRS, helping the medical practitioner understand when and where the helmet needs adjusting throughout the procedure.

1 Idea

Today, in the field of medicine, examining and diagnosing brain diseases for babies and toddlers presents a real challenge. These patients are not very cooperative with traditional methods of diagnostic imaging, since they move frequently and cannot always listen perfectly to directions. However, new developments in the fields of neural networks and deep learning offer promising tools that hold the potential to simplify diagnostic procedures of infant brains. fNIRS, which offers a potential alternative to the widely used fMRI, is overall rather accurate. However, it has one significant drawback – in order for it to work properly, the patient must sit or lie completely still. Thus, it is extremely difficult for the attending medical staff to place it properly at the procedure's start. Throughout the procedure, they inevitably will need to adjust the helmet continuously in between imaging, making for a lengthy and tedious process with increased room for error.

In this project, we researched and attempted to develop an automatic method of mapping the fNIRS helmet in a 3D environment using NeRFMM. NeRFMM is a tool that was made to learn complex scenes from a sparse image set, without precomputing or seeing camera angles and placements. For example, it can consider the fact that a color may change when viewed from different angles. Its main advantage lies in its lack of reliance on preprocessing. A central problem with the original implementation[MST+20] was that it required several hours of running on an additional algorithm (COLMAP) that outputs the camera path. This algorithm might err though, which affects the NeRF imaging output.

We attempted various tests and use cases with NeRFMM. The process was typically as follows - we inputted a scene, got a 3D space, then ran a clustering algorithm on the space to discover the helmet's location. The goal was to identify the helmet's location using stickers that are attached to the helmet, which are translated on the imaging as green dots. By analyzing the stickers' location, it is possible to deduce where the helemet's electrodes are located in relation to the stickers. Combining this analysis with the location of another sticker/green dot, either placed on the patient's face or mapped out by a facial recognition algorithm, would help determine the helmet's location with respect to the brain at any given time throughout the imaging procedure. This end goal, while definitely ambitious, drove the research we conducted over the course of the workshop.

2 Execution

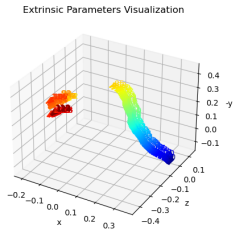
We explored several approaches to tackling this broad and multifaceted challenge. Early on in the process, we discovered that the crux of the issue at hand (found in the 3D spaces) is "ghosting", when different parts of the helmet appear in the imaging out of their correct place. Ghosting significantly impacts our use case, since green dots which don't actually exist could appear on the image, or worse - a green dot may be so split that we can't discern the original at all. Thus, our first order of business was to reduce instances of ghosting as much as possible, at least enough to create a space on which we can easily cluster any stray green dots and still map out the helmet's location. We discovered that almost every run of NeRF's official implementation produced a fair amount of ghosting. Therefore, we tried using a new implementation, called Improved NeRFMM. We found that the central difference between these two approaches was found in the respective neural network layers. In the former approach, the layers were ReLU, while in the latter we found siren layers. This difference led to the "NeRF-MM Improved" implementation yielding improved results.

Another way we attempted to reduce ghosting was by experimenting with the variety and amount of photos used in the imaging. We initialized the algorithm using varying amounts of photos – ranging from a few select video frames to all of them. We found that increasing the amount of photos did not significantly improve the amount of ghosting, especially relative to the broad range of photo quantities we were testing. According to this finding, we can conclude that the algorithm is not heavily influenced by quality. This confirms the premise that NeRFMM is generally designed to work with sparse spaces.

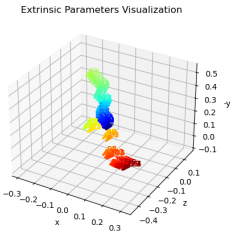
Throughout the research process, we noticed an additional problem – a "jitter" near the end of the learned camera path (Notice the red part in 1). This occurred in each one of our tests, though to varying extents. We theorize that this happened as a result of the camera's slow motion, which led it to change its learned position towards the end.

A common method for solving many NeRFMM-related challenges is Warm Start, which re-initializes the model, using an existing good model trained on similar videos. We applied this solution to our project and got varying results, with the majority showing either few or no improvements in the output. We varied the learning rates in our trials, finding that starting with a high learning rate might end up deleting the learning done by the initial model. On the other hand, starting with a low learning rate could often produce images too similar to the initial model.

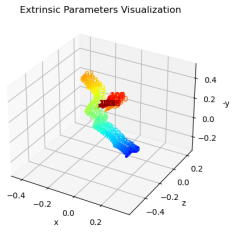
Figure 1: Jitter in Camera Trajectories for Different Videos



Video GX011625



Video GX011613



Video GX011520

3 Clustering

When searching for the clusters of green dots, we noticed that in good models they are both easy to locate and appear clearly on the imaging. Therefore, our clustering algorithm works accordingly – it scans the whole space and identifies green dots of high density. Then, it circles said dots, searching for additional green dots in the vicinity. This method typically worked well - we were able to find a large number of dots in each of the trials.

Beyond this discovery, we moved on to tackle the challenge of locating a fixed bounding box. A bounding box would aid in easier visualization of the helmet as well as in eliminating useless scans. We examined two approaches to locating the box:

1. By manually changing the volume renderer and seeing where it cut off the helmet
2. By searching for the box automatically based on the camera path

While the first method yielded more promising results, we ultimately concluded that a bounding box is not necessarily required in the first place. Our available amount of computing power allows us to search throughout the whole space, without needing to zero in and limit ourselves to a bounding box.

Another crucial aspect of the algorithm lies in its definition of the color green. The color green requires six hyperparameters in the HSV color model. Two of these hyperparameters, S and V (which stand for saturation and value) would be almost trivial to disable, since the goal is to produce bright and colorful dots. This leaves us with four hyperparameters to potentially disable. Generally speaking, the hyperparameters we used do not need much adjustment, aside from when the actual hue of the color is not green. As an aside, we sampled according to different viewpoints, in order to capture green dots from different perspectives. Since in the color in our videos does not change based on point of view, it could be a potential sign that NeRF-based algorithms are not the best fit for this challenge.

4 Next Steps

We recommend several next steps, in order to tackle the issues that arose during our research. First, to rid of the jitter – it could be beneficial to omit the final parts of the video clips. This would hopefully get rid of the specific parts of the video with slow camera paths. However, this would mean also getting rid of valuable data. Second, as described above, there are two ways to locate a bounding box. While we opted to attempt the first approach, there is a good chance that with much additional research, one could find a good bounding box with the second approach – by automatically searching based on the camera path. This is likely a better approach since it requires neither manual work nor fine-tuning of hyperparameters. Third, to achieve overall better results, one could change the NeRF algorithm in our specific case, such that the color in each point would not be affected by the point of view.

Furthermore, we found two additional studies that could greatly contribute to better results :

- "Nerfies: Deformable Neural Radiance Fields" [PSB+21]
- "Instant Neural Graphics Primitives with a Multiresolution Hash Encoding" [MESK22]

The former study suggests that one can view the rays described in NeRF as skewed lines (rather than straight lines), in order to represent rigid transformations in the video. This might help, since toddlers tend to move during the fNIRS scan, and the authors present promising results in scenes of that type. While we attempted to implement this algorithm, our limited computing power ultimately deterred us from reaching final results. The latter study focuses on performance, and its results show incredibly quick training of neural networks in graphics. Specifically regarding NeRF, they show that it takes them three minutes to train it. The paper was published very recently; unfortunately, we did not have the time to implement it on our computers. In addition, it appeared to be a significant engineering challenge to implement other NeRF algorithms, such as NeRFMM, on their framework.

References

[MESK22] Thomas Müller, Alex Evans, Christoph Schied, and Alexander Keller. Instant neural graphics primitives with a multiresolution hash encoding. *arXiv:2201.05989*, January 2022.

[MST+20] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. 2020.

[PSB+21] Keunhong Park, Utkarsh Sinha, Jonathan T. Barron, Sofien Bouaziz, Dan B Goldman, Steven M. Seitz, and Ricardo Martin-Brualla. Nerfies: Deformable neural radiance fields. *ICCV*, 2021.

[WWX+21] Zirui Wang, Shangzhe Wu, Weidi Xie, Min Chen, and Victor Adrian Prisacariu. NeRF−−: Neural radiance fields without known camera parameters. *arXiv preprint arXiv:2102.07064*, 2021.