Replicating BARF and testing it in a real-world scenario

by Bob Brockbernd

BARF vs NeRF

Bundle-Adjusting Neural Radiance Fields is a novel approach to generating a 3D model from still images. It is an improvement on NeRF (Neural Radiance Fields). During training, NeRF is shown images with position and ray direction and learns to predict the colour at the ray's end. This network can then be used to render new images from unseen angles and positions. This is done by giving a ray as input and the output will be the colour of the corresponding pixel.

NeRF is clearly a very promising advancement in deep learning but acquiring the right data can be quite cumbersome since we need to have very accurate 3D positions and angles of the camera when taking the pictures. To mitigate this BARF comes into play. BARF improves on NeRF by allowing pictures without positional and directional information, while not only producing the 3D model it also outputs the positional information that was missing from the training data. At first glance, this might seem impossible since predicting the camera positions requires an accurate 3D model and predicting a 3D model requires knowledge about the camera positions. They are co-dependent. However, BARF introduces a smart way to guide these two to convergence.

BARF starts by randomly assigning positional encodings for each image. During training, it applies filters on the positional encodings. It starts with a low pass filter, this only allows for adjustments in the low frequencies. This makes sure that in the beginning we only make coarse optimisations, and makes it resistant to noise. Over time it slowly moves the filter up to higher frequencies, where it locks the coarse positions in place and focuses on fine-grained movements.

Trying out BARF

I wanted to try this BARF out for myself and see if it was already in a state where an average deep-learning enthusiast could create his own 3D model with just some pictures. Luckily BARF provided a helpful GitHub repository, where running barf is made relatively simple and straightforward.
(https://github.com/chenhsuanlin/bundle-adjusting-NeRF)

Before running BARF on my own images I decided to test if I could get similar results with my setup and their data. Since an average deep learner has no access to a high-performance cluster (and neither do I) I tried running it on my notebook, an HP Zbook.

Comparing results to BARF paper

To make this a viable experiment I had to decrease the image size from 400*400 to 100*100, and run a 10th of the iterations. I ran both BARF and NeRF on the “chair” environment. As you can see the error in rotation and translation is a lot bigger in my experiment due to the fewer iterations that have been run. However the view synthesis quality is better in all cases, this can be explained by looking at the decreased resolution. It makes it more difficult to distinguish between good and bad-quality images. Nevertheless, how the performances differ between NeRF and BARF is similar for both the paper and my experiment.

	Rot	Trans	PSNR BARF	PSNR NeRF	SSIM BARF	SSIM NeRF	LPIPS BARF	LPIPS NeRF
Paper	0.096	0.428	31.16	31.96	0.954	0.961	0.044	0.036
Notebook	0.436	2.07	33.63	35.20	0.980	0.99	0.010	0.010

Below there is an example of an input image (left) and an output image (right). Note that the input image was downsampled to 100*100 for training. The output image is made from a new angle that is not in the training set and is thus generated by BARF.

Real-world data

Now that we know how BARF works, I tried to run with the same setup on images I took myself. I took 110 images from a coffee cup and a can of soda. An example of input and output can be seen below.

As you can see the result is not as clear as the chair, but the output image is not totally unrecognizable, when knowing the input. The big difference in performance is probably due to the background not being transparent. Running BARF for a lot more iterations might let it converge better to the actual 3D model.

Please note that when I tried to use the provided “iPhone.py” dataloader and train BARF on my own images I ran into an issue where torch would not get past the initial validation stage. In GitHub there is an issue about this bug. (https://github.com/chenhsuanlin/bundle-adjusting-NeRF/issues/76) To solve this I reverted the last commit made which was pushed on April 6 2022.

All in all, BARF seems to be a very promising technique but is not yet suitable to run on your own hardware with your own pictures.