SfM analysis works by making a 3D model of a scene through taking regular photos from different angles. Using these photos it finds distinct points which are then matched across multiple photos to determine where those points are in a 3D space. The result is a point cloud with the estimated camera positions. The end result is a 3D skeleton of the scene made from the overlap of the images. For SfM I find it most interesting that no other information other than 2D images from different viewpoints are used to reconstruct a 3D structure. NeRF on the other hand learns a scene and generates realistic new views based on it. To accomplish this NeRF trains a neural network to determine at given 3D locations what should appear, what color it should be and how much should appear. To do this it shoots a ray for each pixel, samples many points along that ray, and blends the predicted color based on density. After training NeRF on many photos it gains the ability to render smooth, photorealistic viewpoints. For NeRF I found it extremely interesting that it can create realistic new camera views of a scene even from angles that weren't photographed.

To complete task 2 I chose to use StereoSGBM (Semi-Global Block Matching) as I found it to offer the best accuracy while also being the easiest to run. SteroSGBM works by matching small image patches between the left and right images to find matching pixels. Compared to other methods which either look at each pixel or the entire image this was the most efficient regarding performance. From the different methods I researched it also was smart enough to produce the smoothest and most accurate disparity maps but also was fast enough to run without a GPU. I ran the project Google Colab so this was important for me as I didn’t want to use a method that used too many resources. To convert the disparity map into a 3D point cloud I used OpenCV's reprojectImageTo3D() function with a Q matrix which I discovered while researching SfMs. Using reprojectImageto3D() I was able to transform the 2D disparity values into 3D coordinates and the results showcase this method was pretty effective. I got over 150,000 clean 3D points with proper depth ordering, smooth surfaces, and sharp edges between objects. 85% - 90% of pixels also had a valid depth estimate showing that most of the scene was properly reconstructed.