NeRF Project Documentation

By Nimrod Wynne

This document covers the onboarding project regarding the use of Neural Radiance Fields (NeRF) to generate a 3D scene using images captured from an unmanned drone's flight.
Task Description:
To research and develop a system that can produce an accurate 3D scene from multi-angle aerial drone photography in a short timeframe (less than 10 minutes).

Issues

Upon inspecting the data, prima facie, there are very few non-stationary objects.
This poses a potential issue as NeRF may have difficulty converging when there are non-stationary objects in the scene. More data is required in order to understand the extent, if at any, of this issue.

First Steps

To help with the understanding of Instant-NGP and NeRF, it was necessary to create my own scene through the use of handheld smartphone videos. Creating these NeRF scenes proved useful to understanding NeRF's behaviours. Key video frames were extracted and fed to COLMAP (a structure-from-motion pipeline). COLMAP estimates the poses of the images. This is fed to Instant NGP which builds a NeRF scene.

Metadata

There are two current formats for metadata:

1. The provided XML file describing the camera's features as well as the pose (rot, centre) of each photo
2. The required JSON format that instant-ngp accepts.

The main issue is understanding the format of the required JSON. It appears to be a homogeneous coordinate matrix describing the translations, rotation, and scaling transforms. However, the values seem to scale in a certain way (perhaps relative to the unit cube)?

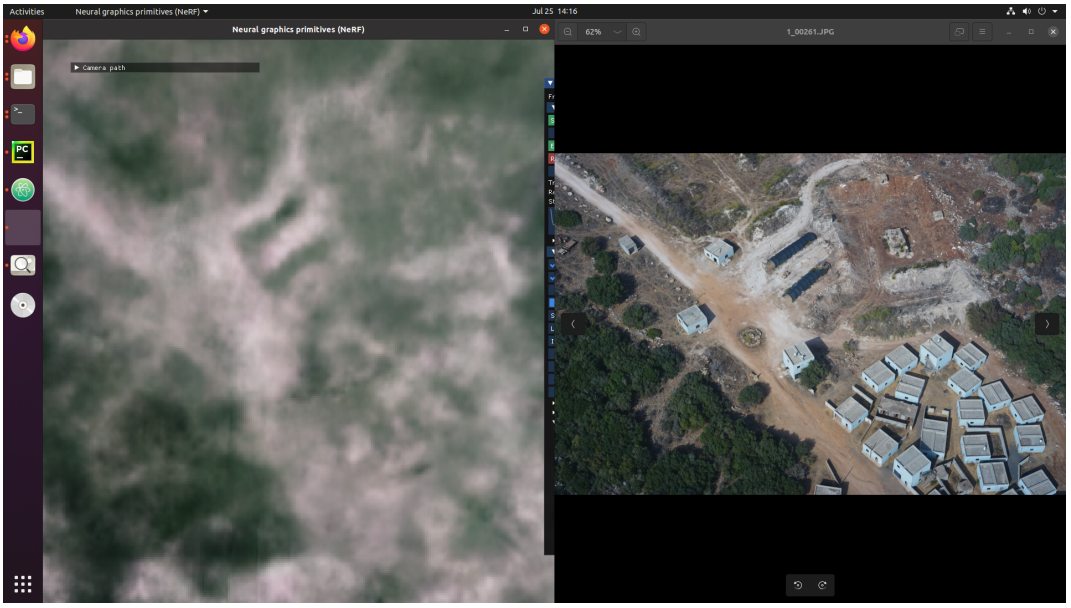
Image Size

The image quality we received in the original dataset is quite large (6000 pixels by 4000 pixels). However, when attempting to load one camera's photos into the GPU (over 500 images) we already reach 5GB of data. Considering how a model of a small scene can easily reach 7GB, our current hardware (Nvidia GTX 1080Ti) does not have enough VRAM to process this much data. The ways to overcome this seem to use either downsampling or skipping images (or both together).

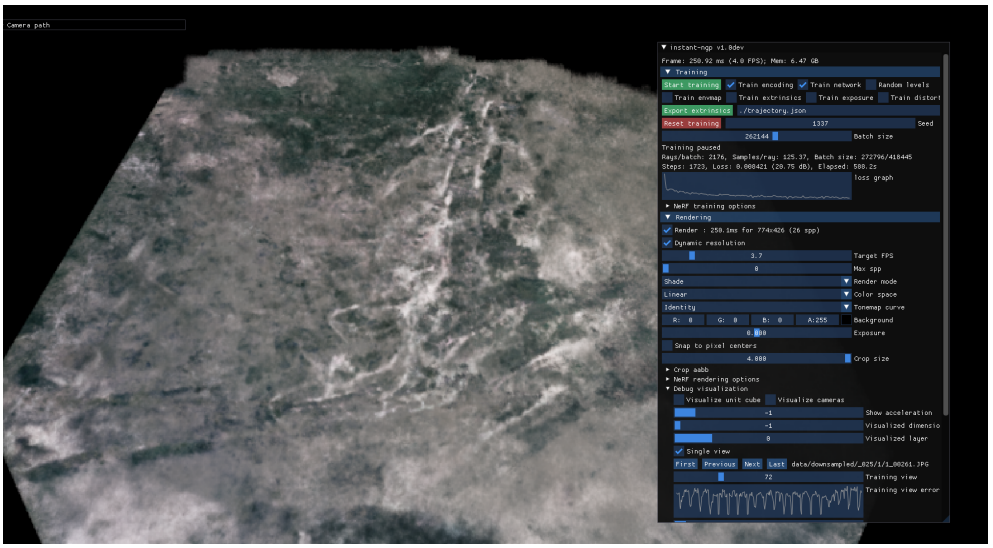
Project Development

NeRF Quality

Currently, the Python scripts that have been written transfer the XML files to the correct JSON format. Nvidia's Instant NGP seems to accept the images and begin training NeRF. However, the 3d model produced is very fuzzy and inaccurate.

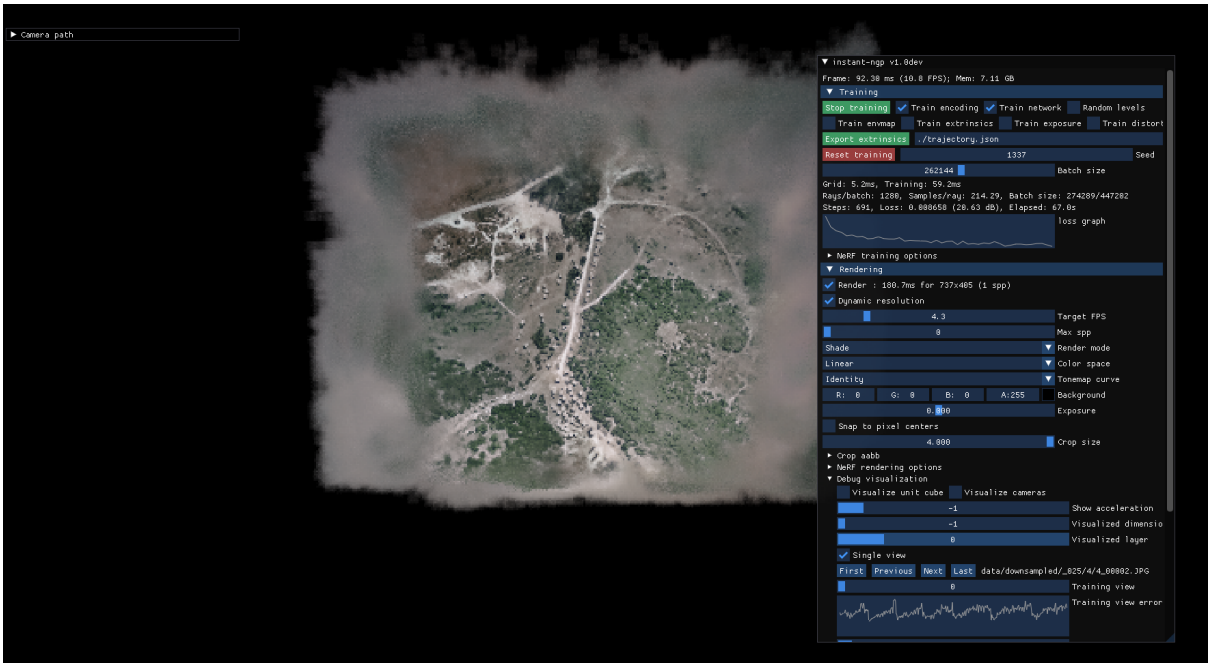


Here is a side-by-side of a NeRF and the corresponding image to that view. We can see that there are similarities in the structure, but the fine details (and colour) are missing. This could be due to errors in the camera's properties (focal length, etc) or because NeRF does not have enough data to recreate the scene properly.



Here we can see an overview, very noisy and inaccurate.

Copying the data generated by COLMAP (only the frame data, i.e. not including sharpness and camera location) gives us this result.



This implies that the issue is not with the camera parameters, although using the focal length from the COLMAP algorithm (1627 instead of 3563) seems to produce vastly different results. A larger focal length (3563) seems to give more depth perception in the model.

https://github.com/NVlabs/instant-ngp/discussions/471

Each Camera does not necessarily have the same camera properties.
New idea:
Perhaps it isn't the transformations that are the cause of the fuzziness in the NeRF image quality. It may be that the GPS data collected from the drone is not accurate enough and when we provide NeRF with this data it cannot build a true scene.
However, it may be that the COLMAP program can more effectively build an accurate 3D mapping for the location of each image than we can through the use of the raw GPS data.

A Mid-Way Recap

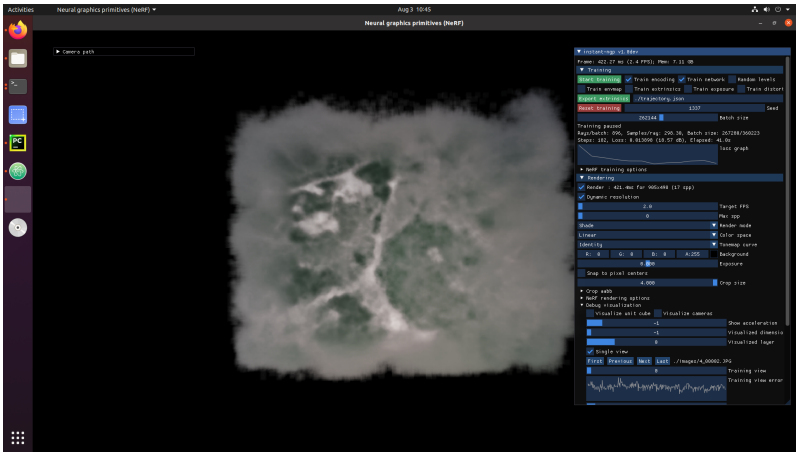
Stopping to examine the work done so far, I am not hopeful that it is possible to recreate an accurate 3D scene through the use of the drone's GPS data.

1. The GPS data may be inaccurate and therefore, when used directly, produces results with plenty of artefacts and distortion.
2. Using COLMAP is too slow, we cannot use COLMAP with 2600+ photos.

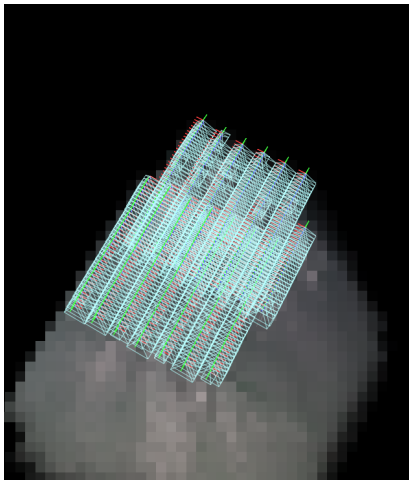
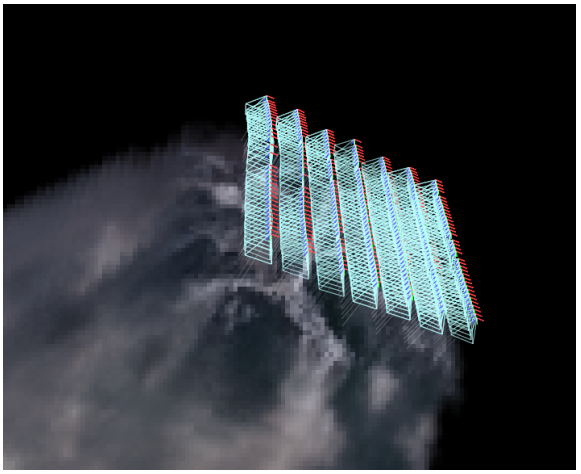
A possible direction could be to use COLMAP for a small subset of images. Somehow, intelligently choose images that view a certain section of the area, and perform the pose estimation on a much smaller subset of images.

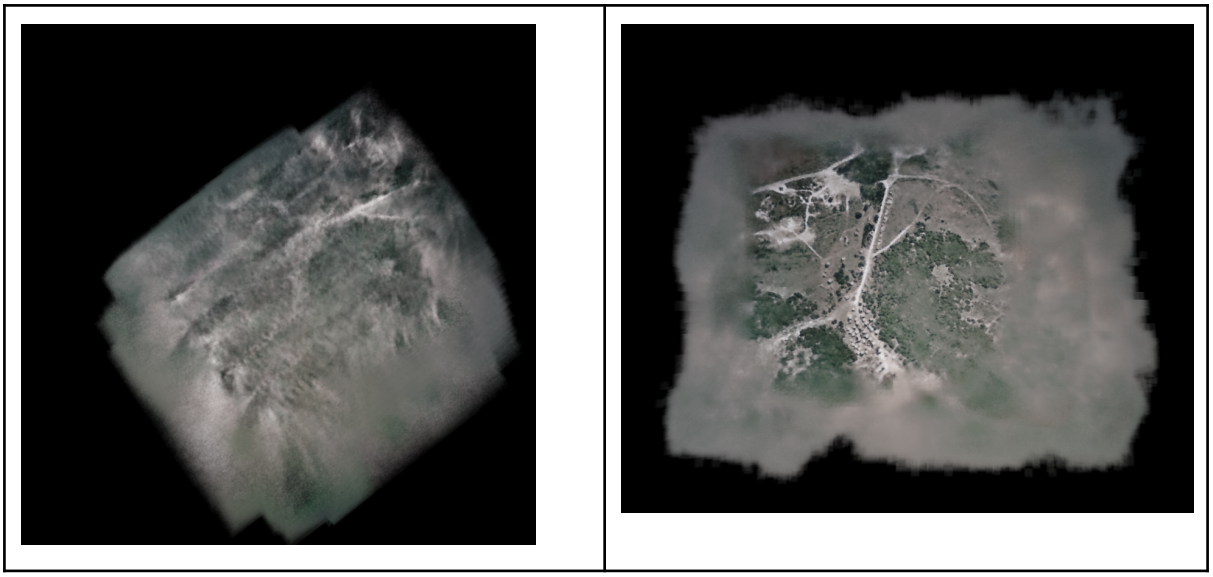
Challenge of GPS Data

A potential source for the noisy and incoherent NeRF scenes could be the GPS data. Due to noise and inaccuracy in the GPS readings, it is possible that NeRF cannot create an accurate 3D scene. Below is an example of the data generated by COLMAP with random normal noise (mean=0, std=0.02) applied to each image's transformation matrix. We can see that adding noise to "perfect" data renders the image very blurry. From this point on, we assume that the artefacts we see are due to inaccurate GPS readings.

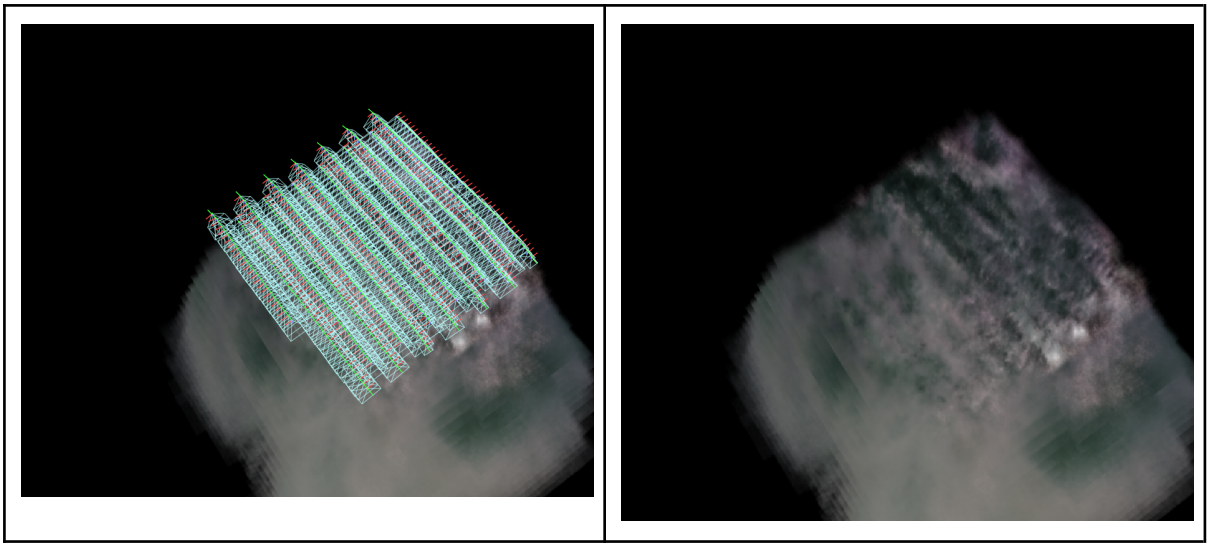


Here we show the difference between the extracted from the drone's XML file, versus what we see when we run COLMAP on the images.

XML DATA	COLMAP GENERATED DATA
	



On the left, we see the scene produced by the drone's XML data. We see that the cameras' positions in the visualisation are not logically sound, since in reality the overlap of the image positions did not look like so. Further, we can see how the final NeRF scene has many artefacts. On the right, we see how COLMAP has positioned the images in a way that makes more sense. However, one may notice that the scene produced by COLMAP does not contain any of the sought-after 3D features.
Below we can see the X, Y positions of the drone as it captured images.



Curiously, when the image positions do appear to be close to the ones generated by COLMAP, the output is even worse than when the XML data did not appear to imitate the drone's positioning.

Testing Individual images

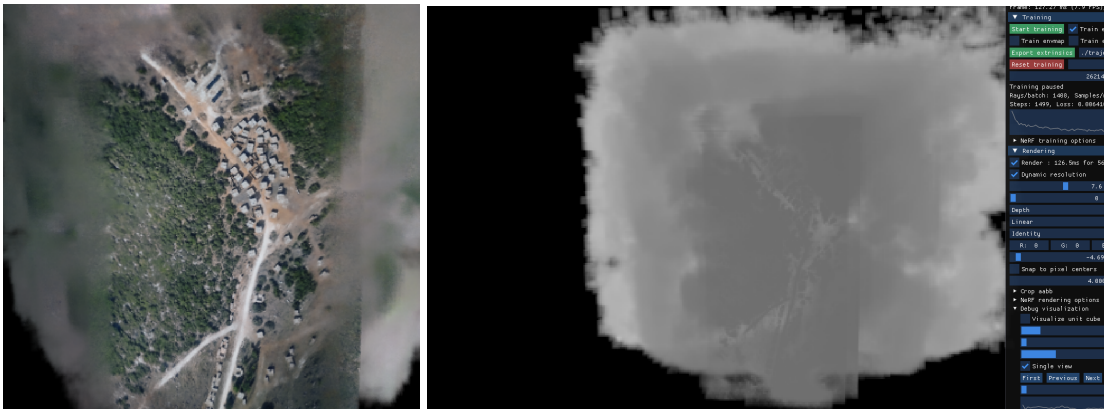
Above is a NeRF scene created from only one drone image. We are looking through the view of the camera. We can see that the image has been recreated very accurately.

In an attempt to diagnose the problem, I prepared a scene with only one image. When we move the camera to "look through" the image's viewpoint, we see that it manages to recreate the picture well. This hints to us that the artefacts may not be a product of camera properties (distortion, focal length) but that NeRF is not able to coordinate the different camera poses to construct a scene.

Trialling a Smaller Region

Single Angle (Down-Facing)

I attempted a different approach. Instead of using all 536 images from the down-facing camera angle, I fed Nvidia's colmap2nerf script only 136 images (from the down-facing camera angle) of the built-up area in the area that was filmed. This took significantly less time (around 15 minutes with COLMAP) and produced a high-quality result. However, there still is very little difference in depth. The scene is interpreted as a near-flat image as opposed to the interactive 3D scene that we want to achieve.



These images are all from the down-facing perspective, purportedly, due to the lack of variation of the camera's angle.

Multiple Angles (Down, Front, Back)

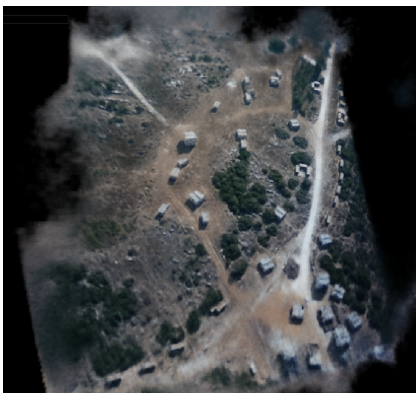
Photos of a built-up area were hand-picked. These photos were chosen as they all visually describe the built-up area from different viewing angles.

Below, we can clearly see that NeRF has generated a 3D scene where the buildings have depth and can be viewed from different angles.
It demonstrates the following points:

1. It is possible to generate a useful 3D scene from the drone images provided
2. The camera's pose estimation works well and quickly using COLMAP. The program managed to generate all the camera's details in under 3 minutes.

This raises the following questions:

1. How can we develop a system that:
    a. A user can select a (small) area that the want to generate a 3D scene of
    b. Intelligently select the images that cover the region selected with a high variation in angles.
2. COLAMP2NERF script seems to behave rather strangely sometimes. We would need to edit this code to ensure that we understand and can predict how it runs each time.
3. How can we improve the quality?

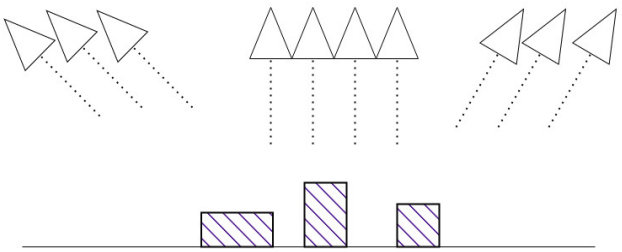


Viewing angle of the scene

A different angle, showing how the scene has generate views of the buildings

Note: that the colmap2nerf script seems to reject certain images. The criteria as per these images are rejected to is not known.

An important question: how does one turn this into a “product”.
One of the tougher sides of this process to automate would be to select images that “view” or “look” at a certain part of the scene,



We could use a kind of trigonometric function to see which images can “see” the selected area.

Smart Image Selection

Part of the issue with generating camera poses using COLMAP is the time COLMAP's algorithm takes to extract features, match the images, and generate the poses.

To counter these long processing times, we can develop a smart way of selecting which images we want to use. Here, the accuracy of the GPS is not as important as the precise location is not so important for choosing which images to give to COLMAP.

The idea is to declare an x, y, z coordinate and radius as the "Area of Interest" and check which images view the area. The area of interest is simplified as a 2D circle that is drawn onto the surface of the terrain.
We can then choose all the images with FOVs that "see" the Area of Interest.

General code flow:

1. Calculate image ray direction
2. Calculate x, y at desired z (the height of the surface)
3. Check if image's x, y and FOV radius intersects with the area of interest
4. Add image path to list of images to use in COLMAP

Performance

The program works as followed:

1. All the image details (coordinates and angles) are read into memory
2. An x,y and radius are chosen as the "Area of Interest" that we want to construct a 3d model of. We assume that the z is surface level
3. Calculate the "circles" that each camera sees at the surface level and check which ones have sufficient overlap with the area of interest
4. Move these selected images to a separate directory
5. Perform Instant-NGPs COLMAP function on these images to extract camera poses
6. Run NeRF

	Here is an angle produced by the program. The program recognised 168 usable images. After COLMAP processes these images, only 45 remain. COLMAP only chooses images from a specific direction. Although the image quality from this direction is fairly good, from the reverse direction it is quite blurry.

Simulate Images' Field of View (FOV)

Finding the Images' Direction Vectors

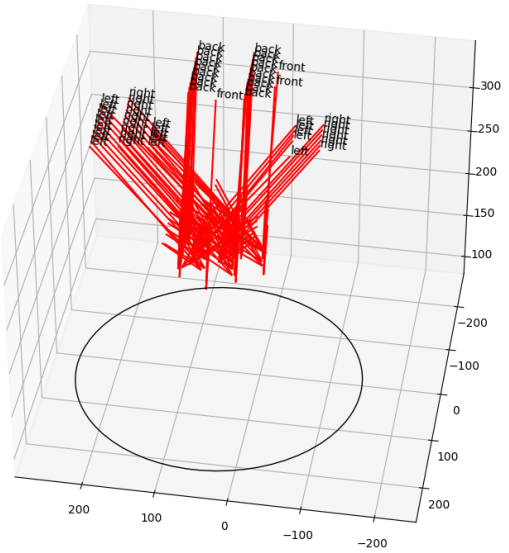
To begin, we first use the rotations provided from the drones metadata. These come in the form of Omega, Phi, and Kappa values. These are commonly used to express the attitude of an aerial object.
To make use of these attitudes, we first construct rotation matrices in the x, y, and z axis (omega as x; phi as y; kappa as z). Then we construct a complete rotation matrix by multiplying all three matrices together:
Rot_matrix = rot_x @ rot_y @ rot_z

```
k, phi, w = self.get_rot()

w = w * (np.pi / 180.)
phi = phi * (np.pi / 180.)
k = k * (np.pi / 180.)
rot_x = np.array([[1, 0, 0],
                  [0, np.cos(w), -np.sin(w)],
                  [0, np.sin(w), np.cos(w)]])
rot_y = np.array([[np.cos(phi), 0, np.sin(phi)],
                  [0, 1, 0],
                  [-np.sin(phi), 0, np.cos(phi)]])
rot_z = np.array([[np.cos(k), -np.sin(k), 0],
                  [np.sin(k), np.cos(k), 0],
                  [0, 0, 1]])
rot_m = rot_x @ rot_y @ rot_z
```

This gives us a homogenous matrix. We then normalise the matrix so that we can calculate the unit vector.

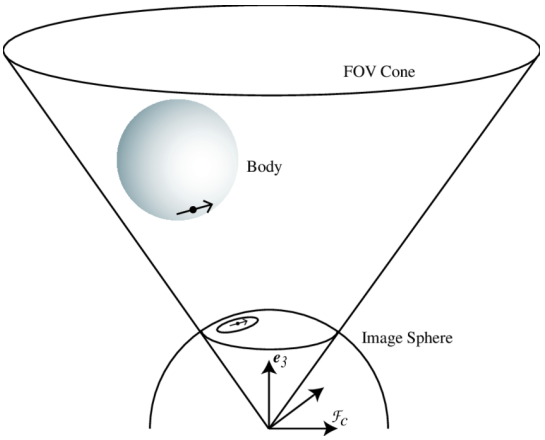
To find the direction vector in three directions (x,y,z), we multiply the homogenous matrix by the vector (0, 0, -1). This creates the vector relative to the camera looking “down” the z axis.



The image, above, visualises the images chosen that view the area of interest (the circle drawn on the graph). Each arrow indicates the image’s direction.

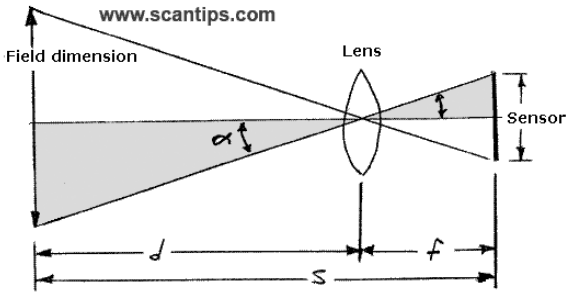
Selecting Images Which “See” the Area of Interest

To start, we can simply draw a ray from the camera and check if it intersects a circle drawn at a particular x,y,z. However, this leads to few cameras being selected as the images are positioned in a sparse configuration and the chances of a single ray intersecting the circle are low. Furthermore, the images use perspective and, therefore, cannot be accurately simulated by a ray. A better approach is to model a Field of View (FOV) cone. This is a simulation of the area that a camera can see. We can calculate the area that the image sees on the same plane of the area of interest and see how much they overlap.



We can use a simple calculation to determine the size of the circle we at a specified distance:
Field Width = (Sensor Size * Working Distance) / Focal Length

We can then calculate if these circles intersect one another. If so, we calculate the IoU (Area of Intersection over Area of Union) i.e. The ratio of the overlapping bits to the bits that don't overlap.



Compound Strategy

The lack of quality from "unseen" angles introduces uncertainty as to whether this method for generating 3D scenes can be used effectively. Currently, the scene generated is impressive but is not high enough of resolution to be useful.
Furthermore, the generation of the scene is slightly random and therefore not reliable.

Conclusion

Working on the project for a month has led to the discovery of significant behaviours regarding NeRF and Instant NGP. However, a final product, which produces a reliable and high-quality result consistently, was not reached. In the conclusion we will discuss what was successful, which issues the project has suffered from (and why) as well as what steps may be taken in the future to build on the current work.

NeRF and Instant NGP

NeRF and Instant NGP have proven to be impressive technologies. Their main advantage being the ability to produce very high quality results from very few images in mere seconds. Instant NGP's easy-to-use tools provided an amazing testing ground. From the 3D scene viewer itself, to the scripts written in order to prepare data from input (e.g. colmap2nerf.py). Learning how NeRF displays its output was an important step in understanding how to produce a good scene. Creating one's own NeRF scene is rather straightforward, still frames are extracted from an MP4 video (usually at a rate of 2 frames per second). Instant-NGP then runs colmap2nerf.py script. This script performs feature extraction and mapping on the images in order to estimate the poses of each image. This is necessary as NeRF must also receive the x, y, z coordinates of each image as well as the "looking direction".
After this process is complete, we can train a NeRF scene in under 45 seconds.

GPS Data and an Alternative Method

The original aim of the project was to use the GPS metadata provided as input for NeRF. This came in the form of x, y, and z positions as well as Omega, Phi, and Kappa values (roll, pitch, and

yaw). The data did not prove useful as the GPS data appears to be too noisy/inaccurate to build a high-quality scene. The data provided has an error of 1-2 metres in the x,y direction and a 7 metre error in the z direction.
Due to this error, it was deemed that to continue, the problem had to be approached from a different angle.

It was decided to use COLMAP's pose estimation in order to complete the data that NeRF expects. With many images, COLMAP's pose estimation can take over an hour and a half to complete. Instead of feeding the program all the images we have received from the drone, we select the images that can "see" the area we are interested in and then feed these images to COLMAP to extract the poses.

This strategy produces decent results, however, proves to be unreliable and, to a degree, uncontrollable. Ideally, the GPS metadata should be used, however, the data seems too inaccurate to yield usable results.

Performance

The performance can be split into two sections: speed and quality of scene.

Speed

When compared to traditional methods, this process can be made to run significantly faster. While building a scene, all the processes can take under 5 minutes, significantly faster than using other, more traditional methods (such as Agisoft Metashape).

Quality

Agisoft Metashape manages to produce a reliable result from a set of images given. The two scenes are of similar quality, however, I believe that with further research into NeRF, NeRF can overcome the quality and fidelity of tools such as Metashape.

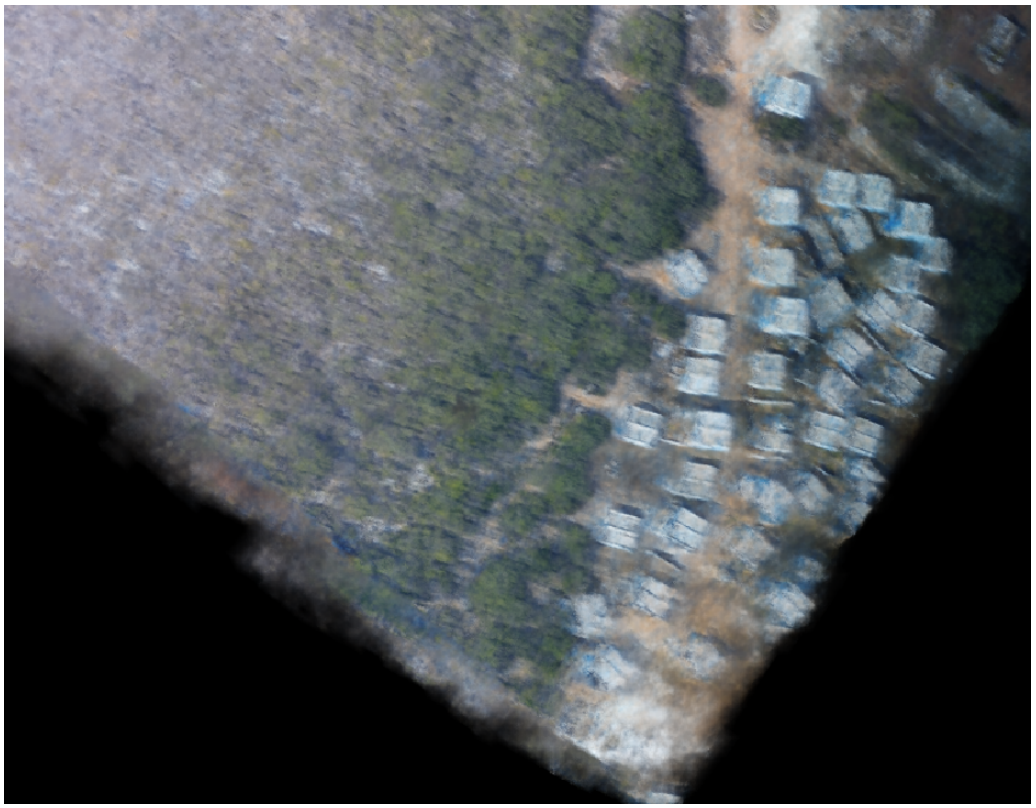
Reliability and Reproducibility

The main issue with the final state of the project is that the output can be unreliable. COLMAP's pose estimation can be treated as a "Black Box" algorithm, i.e. it is hard to predict what its output may be. Furthermore, COLMAP also seems to avoid picking some images that appear to be useful in creating a high-quality scene. It often tends to stick to images from the same general

location in 3D space. This helps build a good resolution scene from the angles close to the chosen images, however, the image becomes blurry and noisy if we view the scene from an angle not described by the images.
Furthermore, we do not yet know the expected behaviour of each scene and each set of images. As shown in the testing stage, sometimes the scene will not converge on a coherent image. This is not a desirable trait and effectively halts the current state of the project from being immediately useful.

In terms of the quality of the scene, we have been able to see very promising results. Below is a comparison of a Ground Truth image and the scene that has been generated by NeRF.

Ground Truth	
NeRF (Instant NGP)	

Agisoft Metashape (Same Angle)	
NeRF (Instant NGP) (Bird's Eye View)	

However, due to COLMAP's method for choosing images, angles that stray too far from the given angles. Theoretically, this would be solved by utilising more images, however, without having full control over COLMAP's pipeline, this proves difficult.

Hardware and Performance

Computer Specifications

System Name	bertie-X299-UD4-Pro
Operating System	Ubuntu 20.04.4 LTS
CPU	Intel® Core™ i7-7820X CPU @ 3.60GHz × 16
RAM	125.5 GiB
GPU	NVIDIA Corporation GP102 [GeForce GTX 1080 Ti] 11GB VRAM
CUDA	V11.7.64
Nvidia Drivers	NVIDIA-SMI 515.65.01

Although the NeRF scene (with the help of Instant NGP) manages to converge in under 45 seconds, viewing the scene with this kind of hardware proves difficult. I recommend upgrading the GPU to a more modern version (RTX 3090). This would enable the scene to be rendered as high quality as possible with decent performance. Note, the current GPU is on its way to becoming obsolete and an upgrade in the near future should be considered.

Further Work

To develop this project into a working product, further efforts are required. Ideally, I would have liked to build a complete product prototype that could be tested by an end-user. However, due to limitations in project duration, I was not able to reach this stage.

COLMAP & Instant-NGP

Delving into COLMAP and Instant NGP's behaviours under certain conditions is essential to building a reliable product. COLMAP currently behaves as a "black box" algorithm meaning that we cannot know exactly what causes certain outputs. Until COLMAP's behaviours can be understood, we cannot build a reliable system.

Methods to improve GPS accuracy

Methods such as Real-Time Kinematic (RTK) and Post-Processed Kinematics (PPK) can be used to correct GPS data and improve its accuracy. RTK drones require a base-station and other pieces of equipment in order to process data in real-time. PPK drones provide more flexibility in terms of the actual flight meaning more freedom as to where the drone is deployed. Furthermore, a PPK drone can refer to past and future data to improve its accuracy.
If accurate flight position data can be collected, the need to use COLMAP to estimate the poses of each image could be eliminated, speeding up processing times.

Test NeRF on More Data

This project was developed on a single drone flight in which there were no significant non-stationary objects in the scene. More drone flights are required in order to assess the project's robustness.
Desired criteria for further tests:

- Light levels (Morning, Midday, Afternoon, Evening, Night). How does the system perform with different lighting conditions?
- Stationariness. Do objects (people, vehicles, etc.) move while the drone is capturing images of the area? How does this affect the quality of the NeRF scene? Is NeRF able to converge?
- Built-up versus Terrain. Can the system perform just as well in generating a 3D scene of topographical terrain as it can generating a scene of a built-up urban area?

Complete Software Package and Field Test

Once a reliable prototype is built, the next step would be to package the system into software for an end user. The user should be able to generate a 3D scene without being exposed to the underlying program and calculations. In theory, the package should display an Orthogonal Map of the area scanned and require the user to choose a region to generate a 3D scene of. All the processing and calculation should happen in the background.

This image, above, is generated from the photos the drone captured. OpenDroneMap produced this orthogonally corrected image.

The Front-End system should allow the user to choose GPS coordinates as their inputs. The system should be able to convert between global GPS coordinates and the drone's X, Y, Z coordinates. This way a user can interface with the system as she/he would any other map. Currently, Instant NGP is designed as a tool for debugging NeRF scenes. Instant NGP's code must be modified in order to simplify the interface. This is possible as Instant NGP is currently open source and available on GitHub.

Finally, testing the system with the desired end-user is a necessity. Direct user feedback is required to understand whether the system fulfils the operational requirements of the user.

Appendix

Testing

To test the Smart Image selection program, we run first the smart selector, then COLMAP on the images it has selected. If COLMAP succeeds, we then run Instant-NGPs NeRF on the generated transforms. To analyse the output and determine a “result”, we can inspect the quality of the scene and how “complete” it is.
A scene of high quality is one where fine details can be seen and the geometric structure of objects is sound.
A complete scene is one where multiple viewing angles produce a high quality image.

Test Sheet

Test #	Area of Interest (x,y,radius)	IoU Threshold	# Images Output (program)	# Images Output COLAMP	COLMAP Duration	Result
001	50,50,150	0.5	36	20	3m6s	Good result however a lot of noise in the scene
002	50,50,150	0.7	14	N/A	11m18s	COLMAP could not piece together the images.
003	50,50,150	0.6	38	21	3m25s	High quality scene. Still lacking multiple angles on the same scene. Therefore, viewing buildings from unobserved angles reveals a spotty scene
004	50,50,150	0.6	60	13	3m29s	Adequate scene. Limited high resolution of viewing angles
005	50,50,150	0.39	167	17	14m50s	Could not converge on the images' positions. Could not create a scene.
006	50,50,150	0.46	100	39	3885s	Same issues as previous. Not enough variation in the angles
007	50,50,150	0.48	73	14	4m22S	No improvement
008	50,50,150	0.499	38	21	3m2s	Same result. The method for finding suitable images needs to include more variation of angles
101	-50.-120,150	0.486	86	23	6m33s	Same result. The method for finding suitable images needs to include

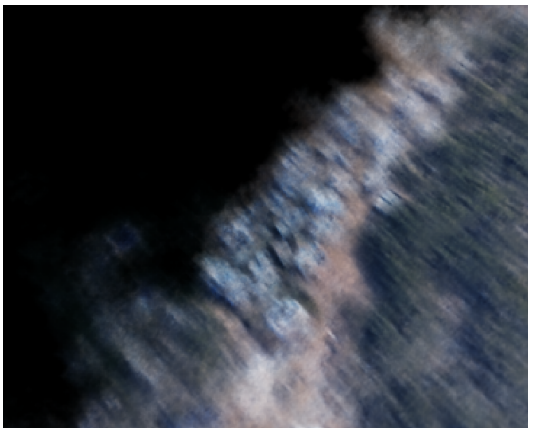
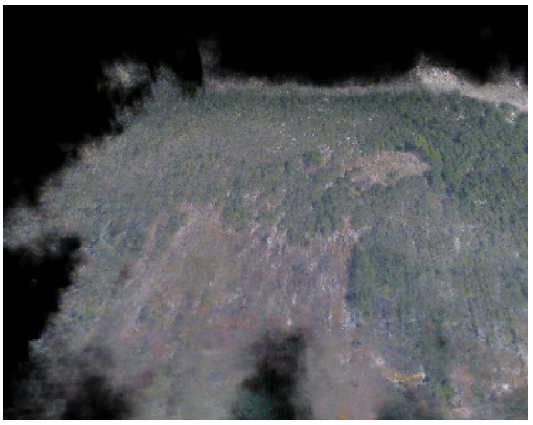
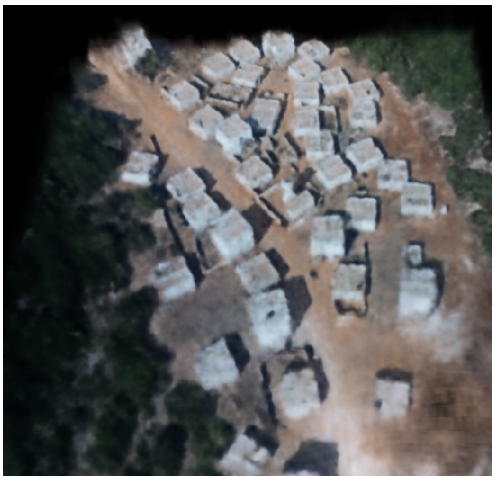
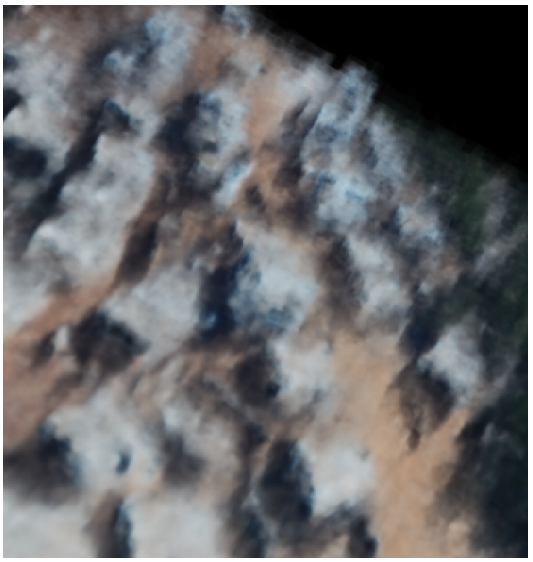
						more variation of angles
102	50,50,200	0.3	230	105	35m43s	Not necessarily a better scene even though more images are used. Resolution of the scene (buildings, etc) did not improve. Too many angles could “confuse” NeRF?

Test #	Area of Interest (x,y,radius)	COMPOUND THRESH	# Images Output (program)	# Images Output COLAMP	COLMAP Duration	Result
201	50,50,150	0.30	89	45	11m16s	Result is good quality although COLMAP still does not choose more drastic angles to help with visualising reverse sides of the scene
202	65,20,75	>97th Quantile	128	66	16m32s	Incoherent mess, it seems like sometimes adding more images may not help

Image Reference

The table below contains images to document how the scene appears.

Test #	Image 1	Image 2
001		N/A
003		
003		

004		
101		
102		

201		
202