# High-D Dataset Visualization

**With the help of Cloud Tools**

Monil Patel 100727400
Soham Bhavsar 100726376
Thayan Sivathevan 100707190
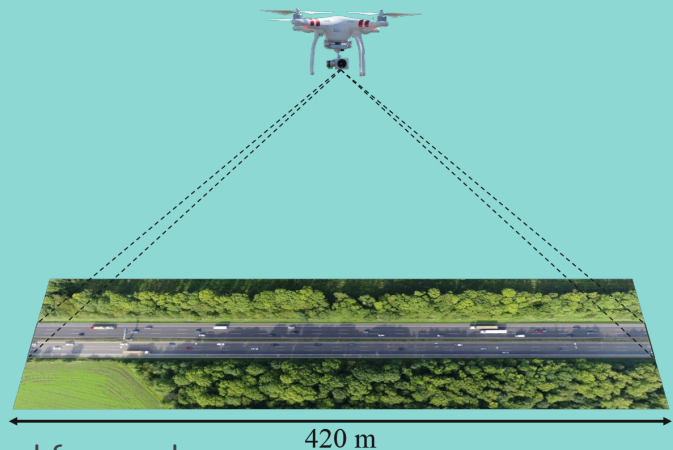Michael Metry 100747141
Rushin Chudasama 100740374

Google Cloud

# Project Scope



420 m

- HighD is a dataset of natural vehicle trajectories collected from a drone
- We want to perform some sort of processing on the dataset using cloud tools
- Our chosen direction was a method of visualization
- We will reverse engineer the data to provide a video of the highway conditions
  - Video can help annotate objects
  - Can be created from simple csv data

# Cloud Tools

- Our main cloud platform was Google Cloud.
    - It was a familiar technology that was used in classroom assignments
    - Our labs helped us to familiarize ourselves with the Google Cloud Platform
- We will use a simple dataflow job to simulate the overall project flow
- We will store and retrieve data from a Cloud Storage Bucket
- Apache Beam will be used to create the data pipeline

# **Methodology**

Create Images Function

- Read the data
- Process data
- Create Plots
- Save images to cloud storage

Create Video Function

- Read image directory
- Iterate over files
- Append image file to video write
- Save video to cloud storage

```python
import os
import shutil
import cv2
import io
import apache_beam as beam
from apache_beam.options.pipeline_options import PipelineOptions
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.patches as patches
from google.cloud import storage
```

```
17    df = pd.read_csv('gs://highd_project/01_tracks.csv')
18    groups = df.groupby('frame')
19
```

```python
for frame, group in groups:
    # Create a new figure and axis for the current frame
    fig, ax = plt.subplots()

    # Set the axis limits
    ax.set_xlim(0, 200)
    ax.set_ylim(0, 50)

    # Iterate over the rows in the current group and add a rectangle to the axis for each row
    for index, row in group.iterrows():
        rect = patches.Rectangle((row['x'], row['y']), row['width'], row['height'], linewidth=1, edgecolor='r', facecolor='none')
        ax.add_patch(rect)
```

```python
for index, row in group.iterrows():
    rect = patches.Rectangle((row['x'], row['y']), row['width'], row['height'], linewidth=1, edgecolor='r', facecolor='none'
    ax.add_patch(rect)


# Save the figure as an image in the local directory
blob = bucket.blob(f'images/{frame}.png')
## Use bucket.get_blob('path/to/existing-blob-name.txt') to write to existing blobs
buf = io.BytesIO()
plt.savefig(buf, format='png')
buf.seek(0)
blob.upload_from_file(buf)
# plt.savefig(f'gs://highd_project/images/{frame}.png')
buf.close()



# Close the figure
plt.close()
```

```python
def generate_video(image_dir, video_path, fps=60):
    # Set up the OpenCV video writer
    fourcc = cv2.VideoWriter_fourcc(*'mp4v')
    video_writer = cv2.VideoWriter('zmy_video.mp4', fourcc, fps, (640, 480))

    # Loop through all image files in the local image directory
    count = len(os.listdir(image_dir))
    for i in range(count):
        image_path = os.path.join(image_dir, f'{i}.png')
        img = cv2.imread(image_path)
        if video_writer is None:
            height, width, _ = img.shape
            video_writer = cv2.VideoWriter(video_path, fourcc, fps, (width, height))
        video_writer.write(img)

    video_writer.release()
```

```python
def run_pipeline(argv=None):
    """Runs the dataflow pipeline."""
    pipeline_options = PipelineOptions(argv)
    with beam.Pipeline(options=pipeline_options) as pipeline:
        # Read the file as a bounded source
        input_pcoll = (
            pipeline
            | 'Read File' >> beam.io.ReadFromText('gs://highd_project/01_tracks.csv')
        )

        # Apply the transformations to generate the images and video
        (
            input_pcoll
            | 'Generate Images' >> beam.Map(generate_images)
            | 'Generate Video' >> beam.Map(generate_video, 'gs://highd_project/images', 'gs://highd_project/my-video.mp4', 60)
        )


if __name__ == '__main__':
    run_pipeline()
```

# Result
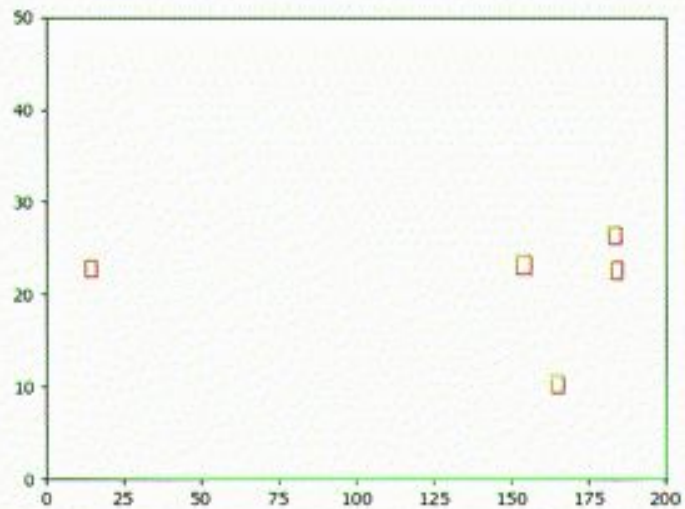
# Conclusions

- The code was able to successfully create a final output video of the data
- We were able to store and retrieve data from our cloud storage
- Cloud tools helped in creating a highly available project, without having to worry about resource management
- In the future, our project will be improved to process multiple csv data files respectively