AUTONOMOUS FLIGHT PLANNING

SAINT MARYS MSBA: S. ZHU K. BRAND C. SUAREZ M. SANTOS

FOREWORD

We want to thank Nelson, Brian, and Navid for their assistance this project, and all of our former professors throughout the program. Additionally, we want to recognize the support and collaboration we received from other members of the Saint Mary's MSBA program. Thank you!

AGENDA

"Drones overall will be more impactful than I think people recognize in positive ways to help society."

BILL GATES

1. Team Introduction
2. The Journey of Cohort 12
3. Business Problem
4. Project Background
5. Our Approach
6. Demo
7. Tech Stack
8. Future Work
9. Questions

THE TEAM

Kevin Brand

Saint Mary's College - System Specialist

Hobbies / Interests:

Sports

Cooking

Video Games

Colin Suarez

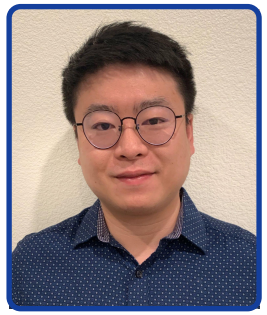
Chiropractic- Office Manager / coaching high school basketball

Hobbies / Interests:

Sports

Graphic design

Travel



Steven Zhu

Walmart.com- Software developer

Hobbies / Interests:

Cooking

Rubiks cubing

Soccer

Mark Santos

Mobility Works- Territory Manager

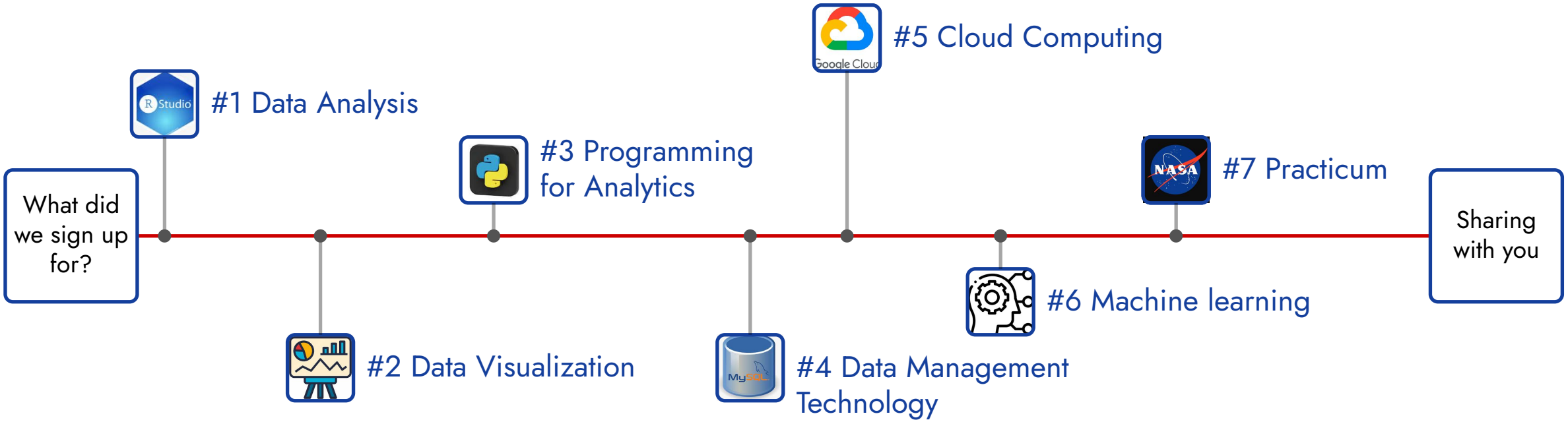
Hobbies / Interests:

Family

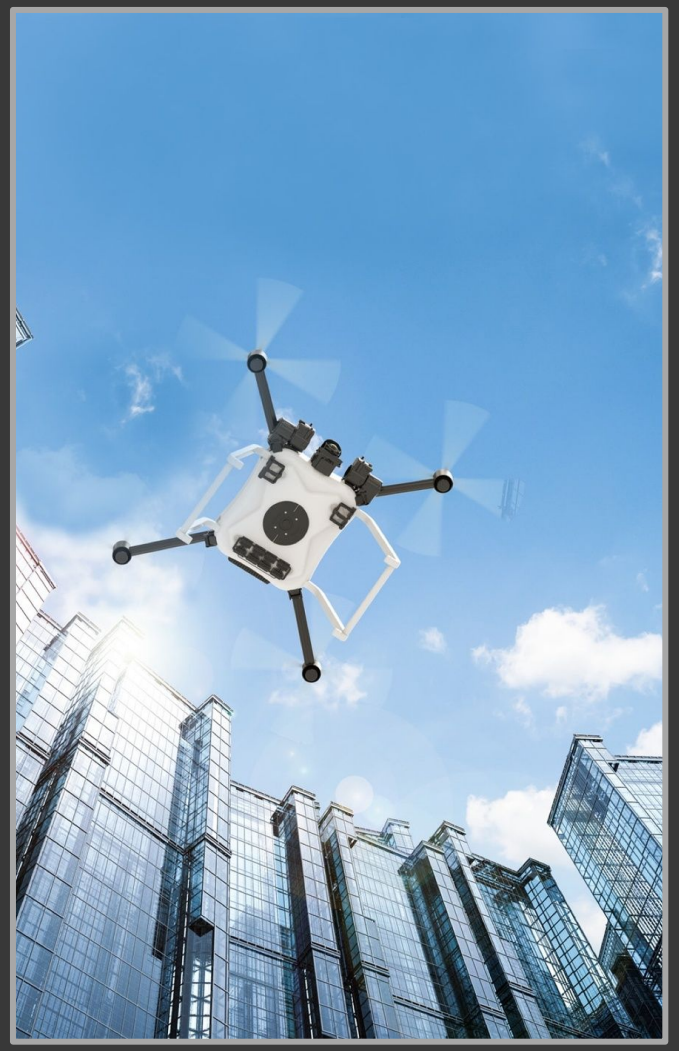
Motorcycles

Mountain Biking

OUR JOURNEY TO TODAY



Business Problem



NASA is seeking to build an obstacle detection and avoidance database for unmanned aerial vehicles (UAVs)

This database would allow drones to fly autonomously to their intended destination, saving time and resources. The scope of our project involves building the obstacle database itself in a fast, accurate and efficient way; what the drone does with this information is beyond what we set out to accomplish, but can certainly be explored upon in future work.

This presentation will take you through the journey our team took to deliver the requested product.

PROJECT BACKGROUND

Why it matters

Why has it not been solved?

Current technology often has limits on the memory and computation power of most drones. The market needs a solution that can be used efficiently and reliably on ALL drones, not just the high end or military spec drones.

Why is this a relevant project?

With the growth done use expected to only increase, so does the need for them to be utilized autonomously and safely. For competitive advantages this must be accomplished while ensuring they are using the most optimized flight path to the destination.

PROJECT BACKGROUND

Market Research

Companies using autonomous drones

- Amazon
- Google (Wings)
- UPS
- Zipline

Value Propositions

- Less pollution and traffic from courier vehicles
- Shorter and more accurate ETA for delivery/scheduled delivery
- Minimizing theft of packages
- Increased accuracy of drop offs
- Delivery of urgent supplies

PROJECT BACKGROUND

Our initial thoughts and questions

Altitude-

Why can't we just program the drones to fly at an altitude that limits risk of most obstacles?

Cameras-

Why are we not using camera technology?

Google-

Why not simply load Google Map view into the drone?

Satellite Images-

Looking at the USGS satellite images, the closest tile is still very far.

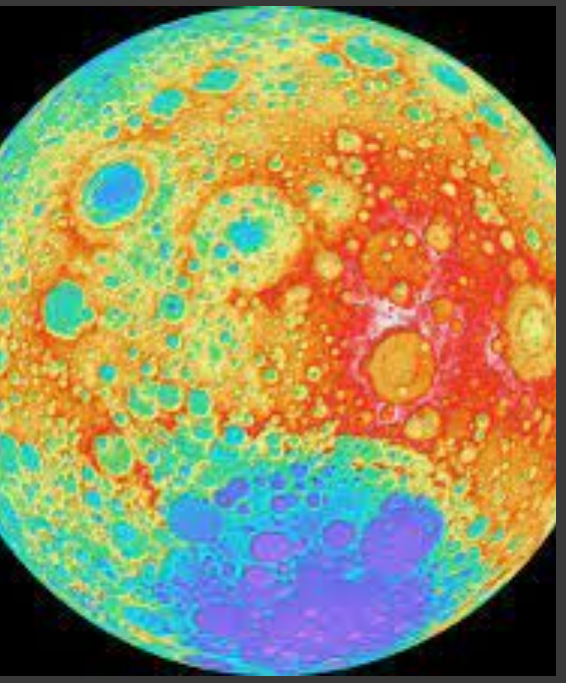
(8500 feet X 3600 feet Tile) Scale 1:9,028

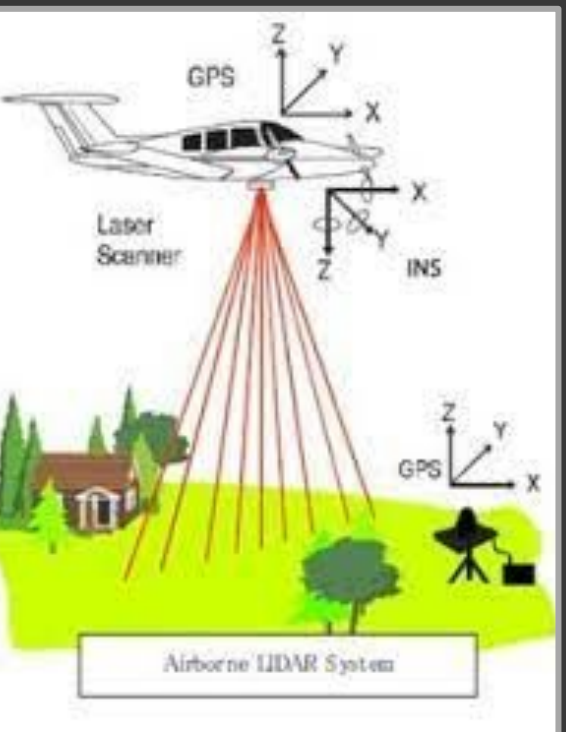
LiDAR-

Why are we considering using LiDAR? If Elon say's its wrong for autonomous vehicle usage, it must be wrong for drones, right?

Data-

Does NASA have access to labeled data?





PROJECT BACKGROUND

What is LiDAR?

Development:

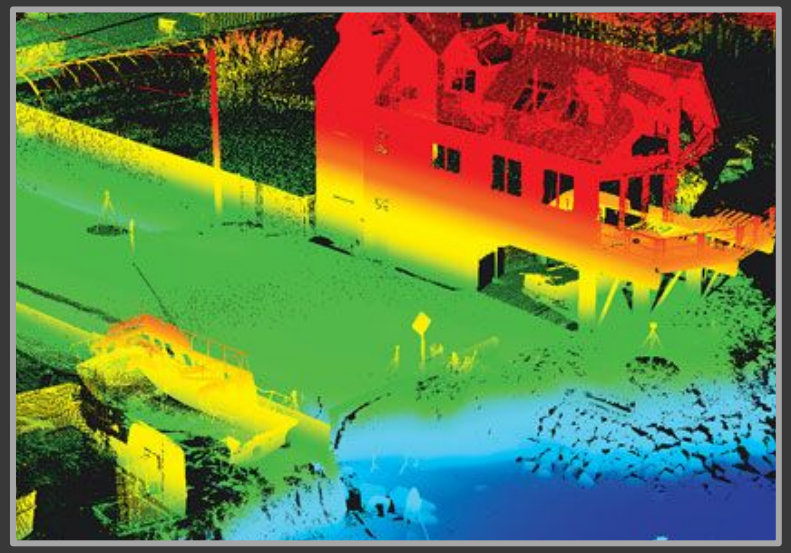
LiDAR is an acronym that stands for Light Detection and Ranging. Lidar is a remote sensing method to examine surfaces

How does it work:

A LiDAR system calculates how long it takes for beams of light (lasers) to reach an object or a surface and reflect back to the scanner. Depending on the sensors used, LiDAR scanning units can fire hundreds of thousands of pulses per second. By completing this in a quick progression the instrument builds up a complex map of the surface it is measuring

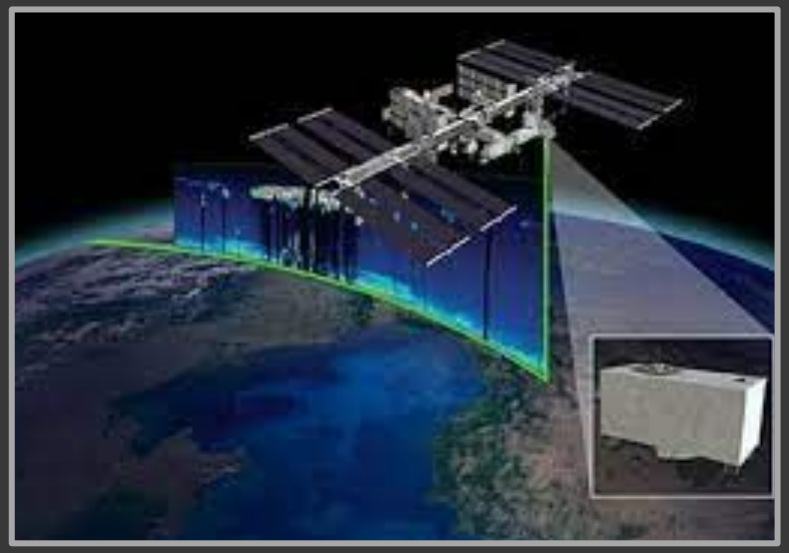
PROJECT BACKGROUND

Types of LiDAR



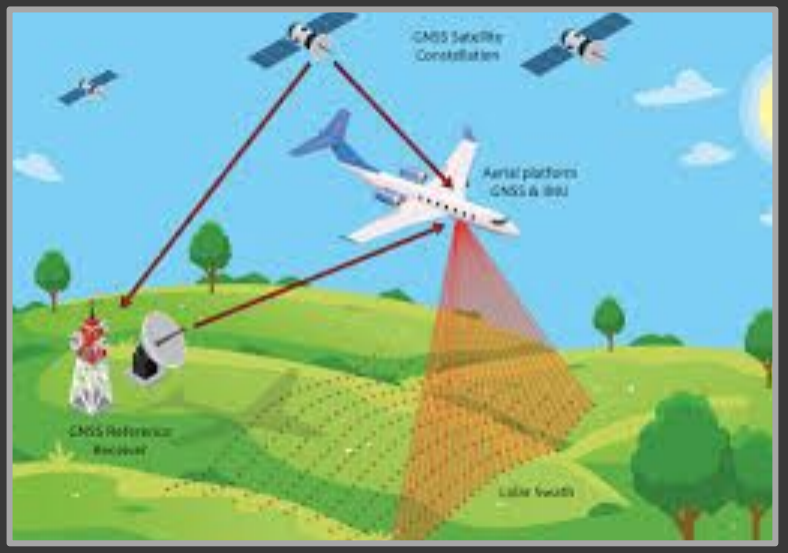
Terrestrial-

- Scanners are installed on a tripod or vehicle
- Applications in surveying and 3-D models
- Can be Mobile or static



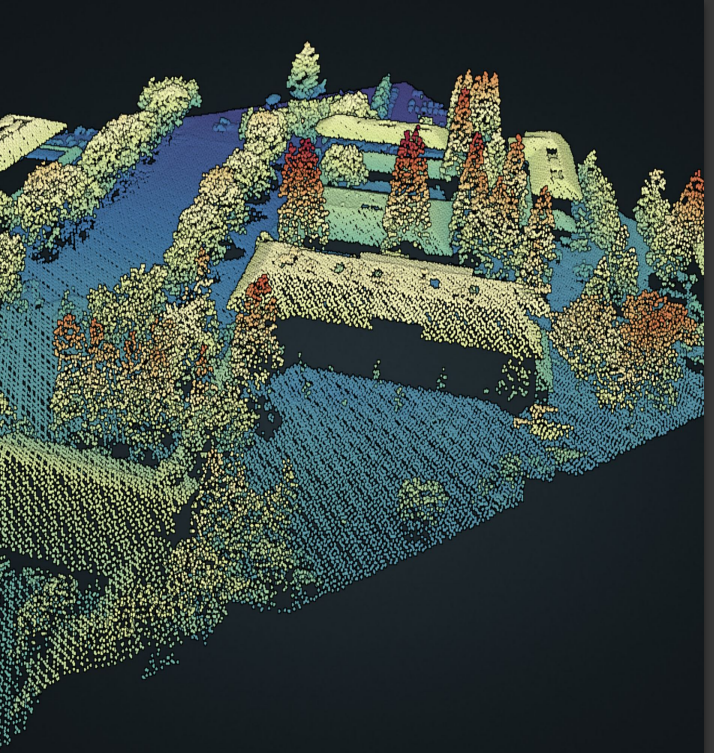
Satellite-

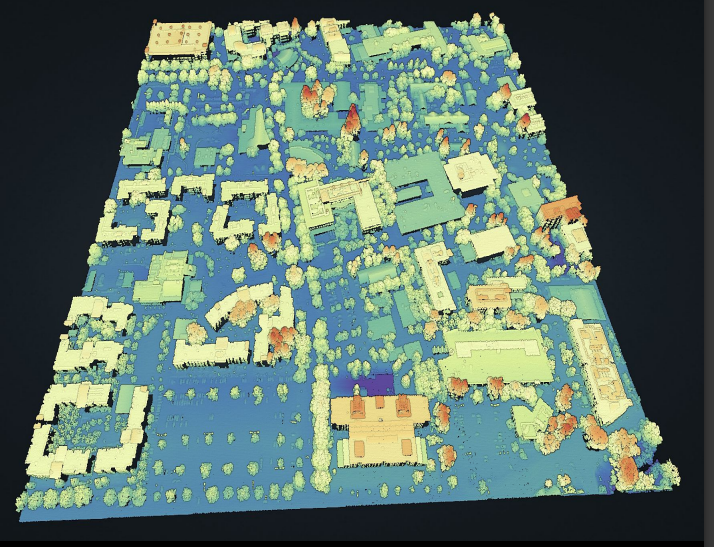
- LITE Launched in the mid 1990's
- Used to produce atmospheric measurements
- In 2006 CALIPSO was launched



Aerial-

- Scanners are installed on aircraft
- Most accurate way to create elevation models
- Scans create 3-D point cloud models of landscape





PROJECT BACKGROUND

Point Clouds

What is a Point Cloud:

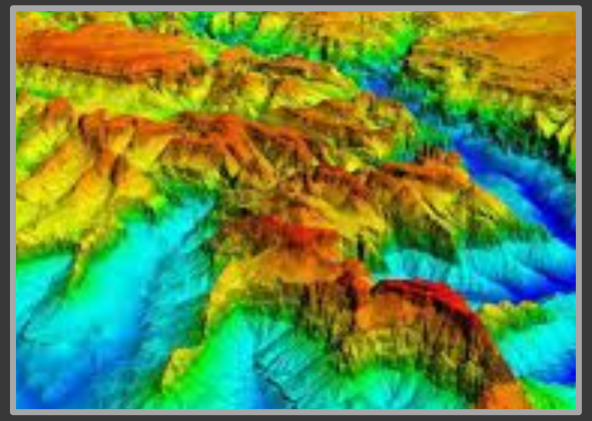
Point cloud data is the term used to refer to the data points collected for a given geographical area, terrain, building or space from an aerial scan. A LiDAR point cloud dataset is created when an area is scanned using light detection and ranging

What is Point Cloud Processing:

Point Cloud processing is a means of turning point cloud data into 3D models

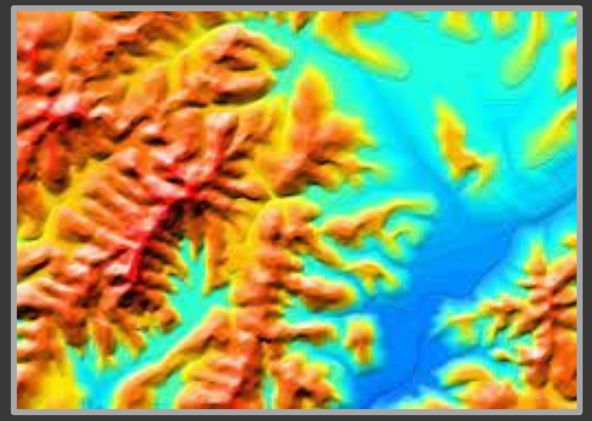
PROJECT BACKGROUND

Common LiDAR Data Models



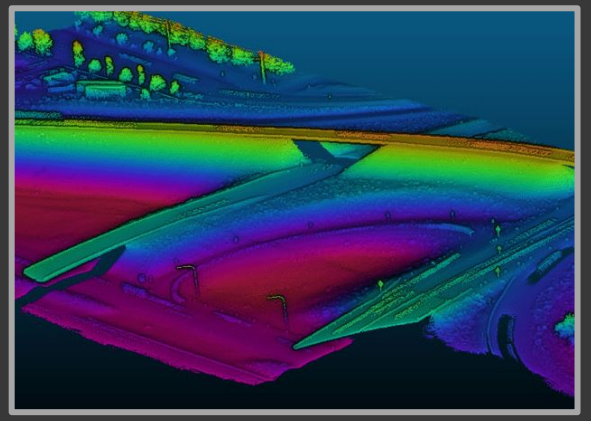
Digital Elevation Model (DEM)

Represents the bare-Earth surface, removing all natural and built features



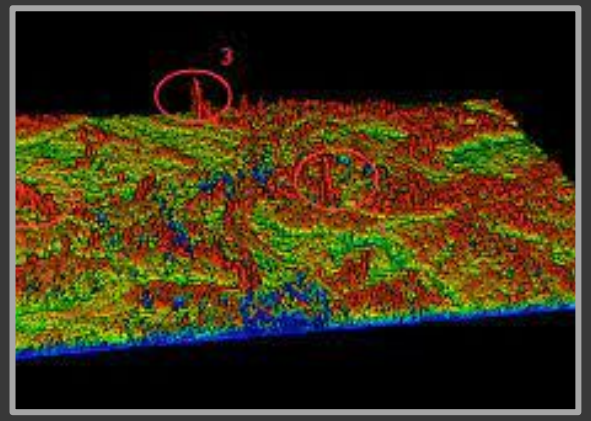
Digital Terrain Model (DTM)

Typically augments a DEM, by including vector features of the natural terrain, such as rivers and ridges



Digital Surface Model (DSM)

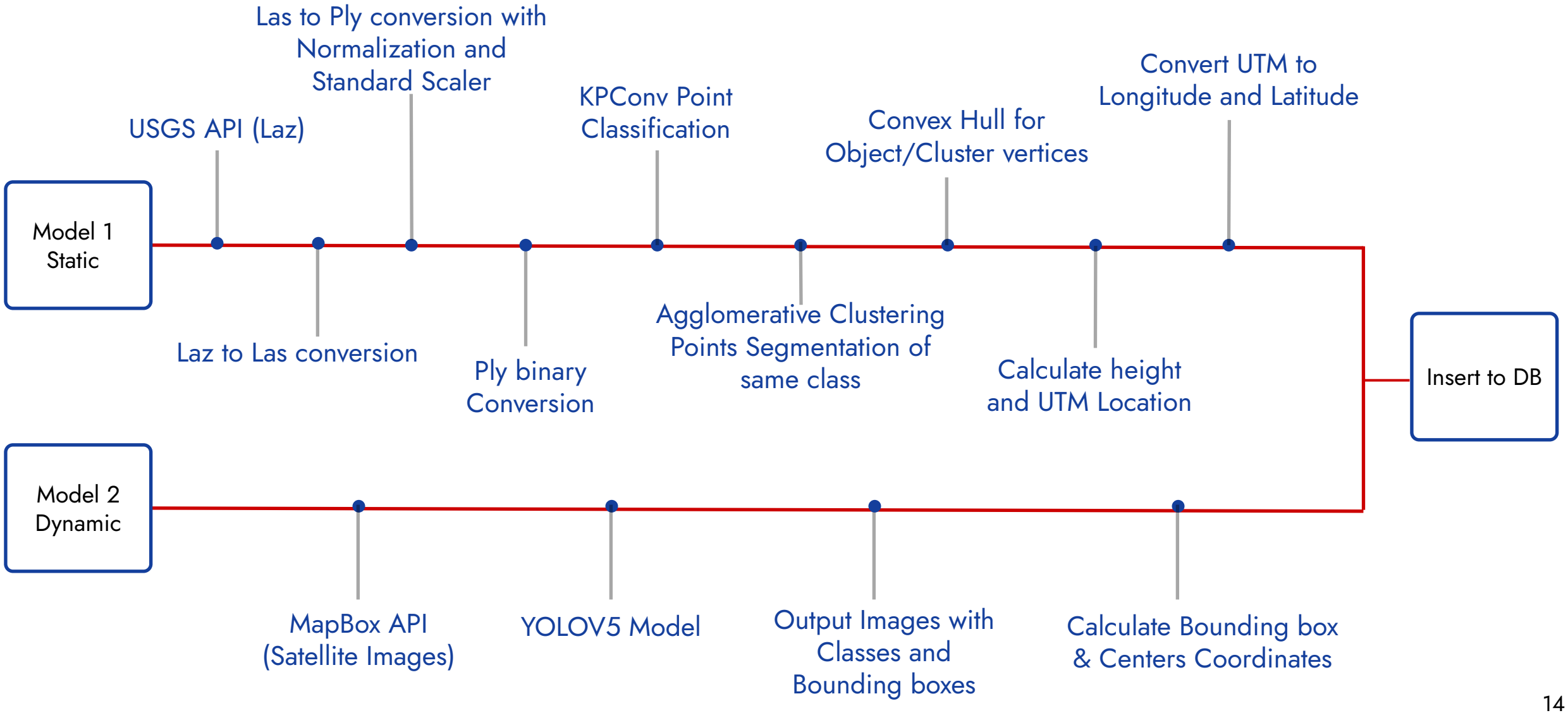
Captures both the natural and built/artificial features of the environment



Canopy Height Model (CHM)

Calculated by subtracting the digital terrain model (DTM) from the digital surface model (DSM)

Overview of Our Approach



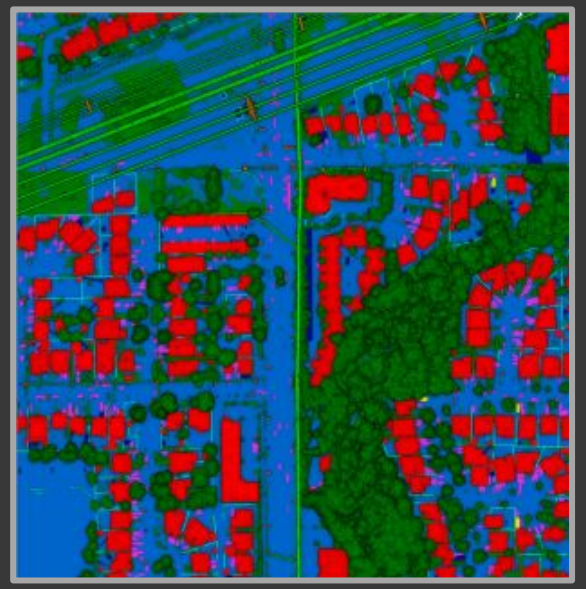
OUR APPROACH

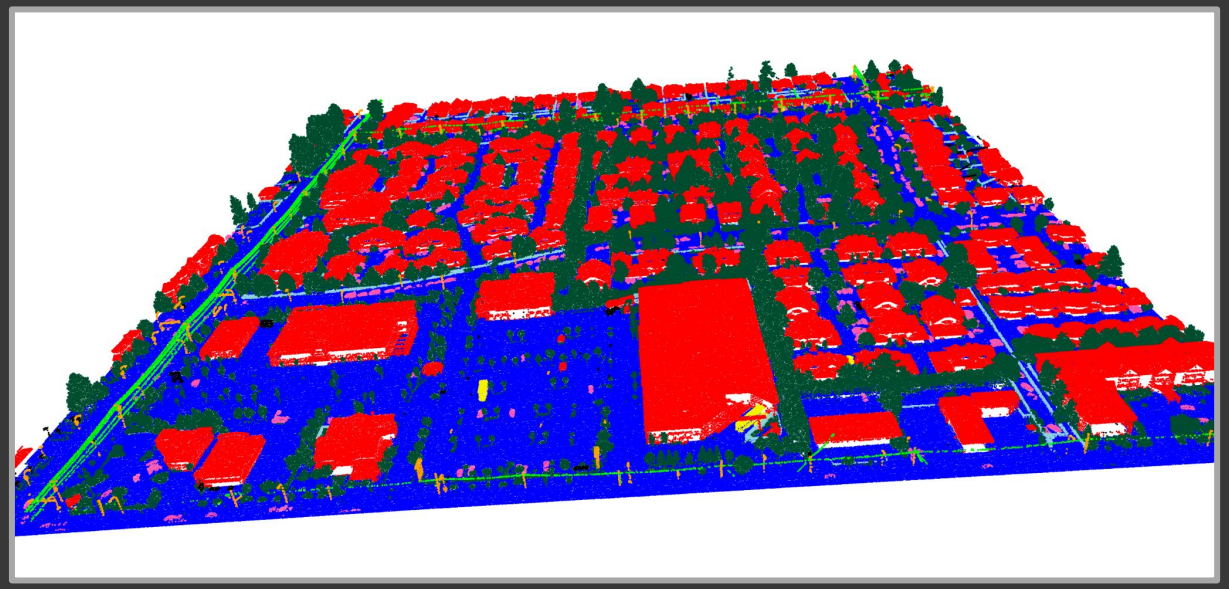
The Data

The **DALES** or Dayton Annotated Laser Earth Scan is a large-scale Aerial LiDAR dataset for Point Cloud Segmentation Developed by University of Dayton

Dataset benefits:

- Contains Over a half billion hand-labeled points
- Pre split training and testing files, cleaned
- Includes two files types: LAS and PLY
- Eight categories: ground, vegetation, cars, trucks, powerlines, poles, fences, and buildings

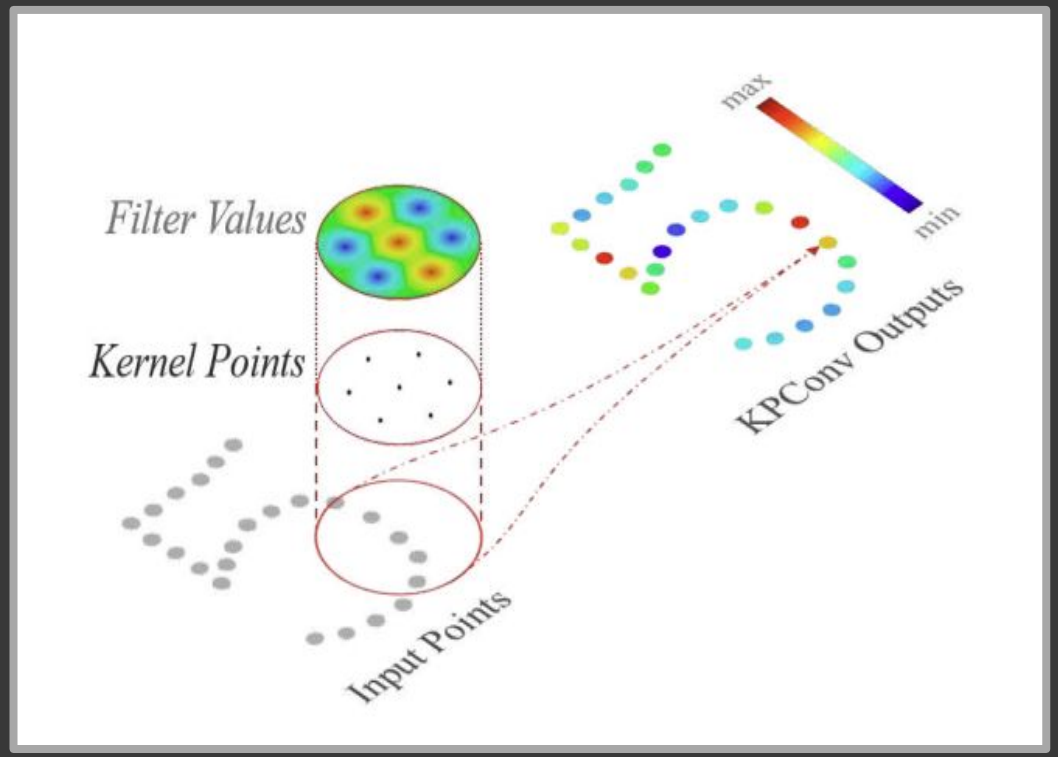




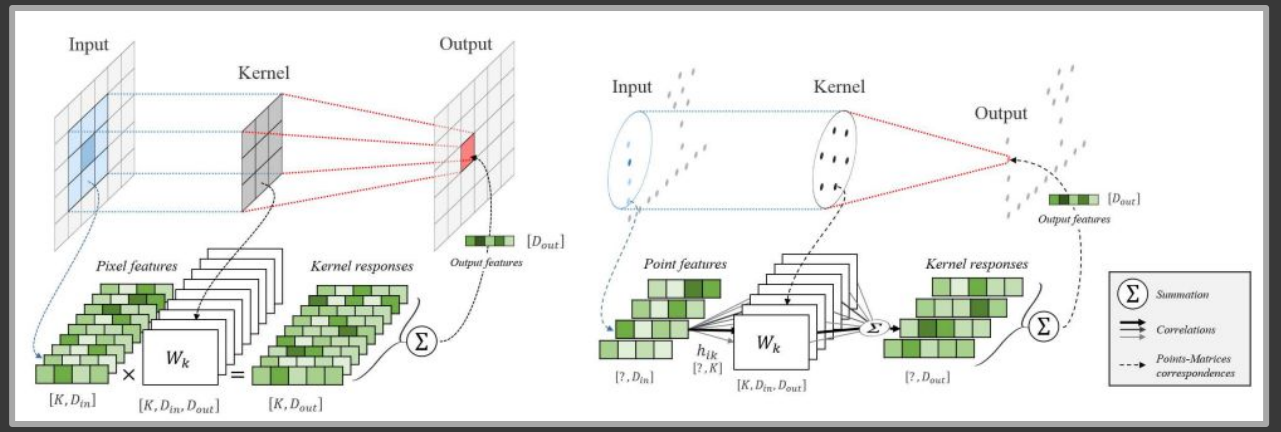
OUR APPROACH

First Model - KPConv (Kernel Point Convolution)

- Newer, more advanced point cloud model
- Efficient and robust to varying densities with it's regular subsampling strategy
- Has built in functions for visualization and metrics to help better understand the model



		IoU								
Method	OA	*mean*	*ground*	*buildings*	*cars*	*trucks*	*poles*	*power lines*	*fences*	*veg*
KPConv [8]	**0.978**	**0.811**	0.971	**0.966**	**0.853**	**0.419**	**0.750**	**0.955**	**0.635**	**0.941**
PointNet++ [18]	0.957	0.683	0.941	0.891	0.754	0.303	0.400	0.799	0.462	0.912
ConvPoint [2]	0.972	0.674	0.969	0.963	0.755	0.217	0.403	0.867	0.296	0.919
SuperPoint [10]	0.955	0.606	0.947	0.934	0.629	0.187	0.285	0.652	0.336	0.879
PointCNN [11]	0.972	0.584	**0.975**	0.957	0.406	0.048	0.576	0.267	0.526	0.917
ShellNet [28]	0.964	0.574	0.960	0.954	0.322	0.396	0.200	0.274	0.600	0.884



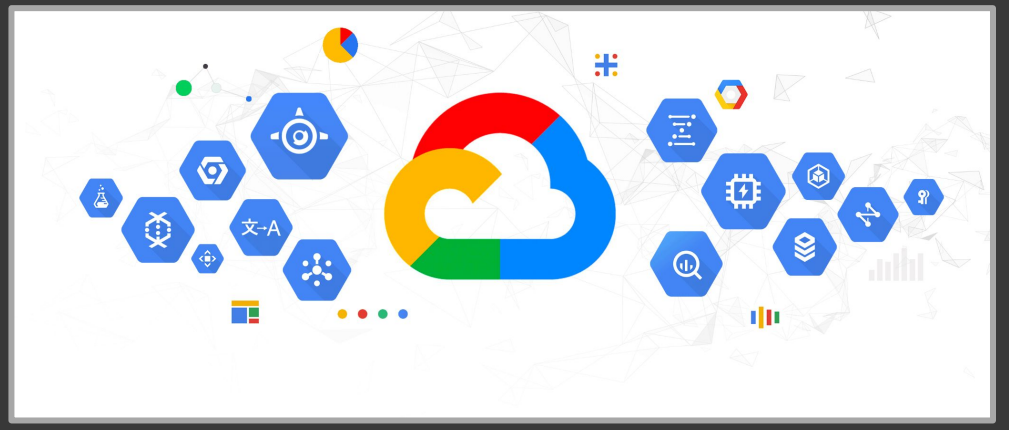
OUR APPROACH

Training KPConv on Dales

- We modified an open source TensorFlow implementation of KPConv to make use of the DALES dataset
- Using the power of the Google Cloud Platform, we set up a virtual machine to train our model with a stronger GPU and more RAM than our local machines
- Code and data were loaded onto the VM
- Training took 6 and a half days to complete a total of 850 epochs
- The overall model accuracy reached 97.35%

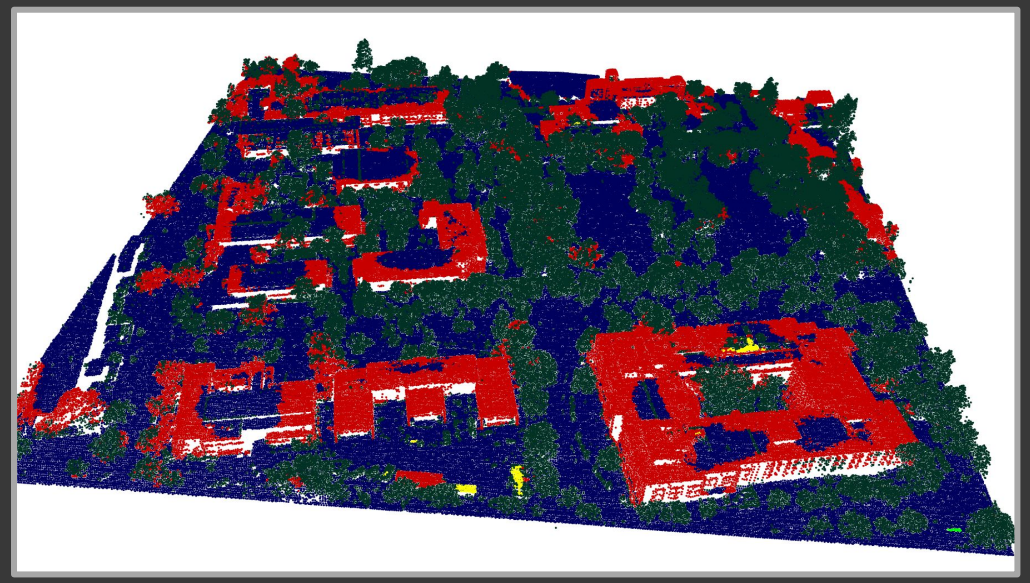
Machine configuration

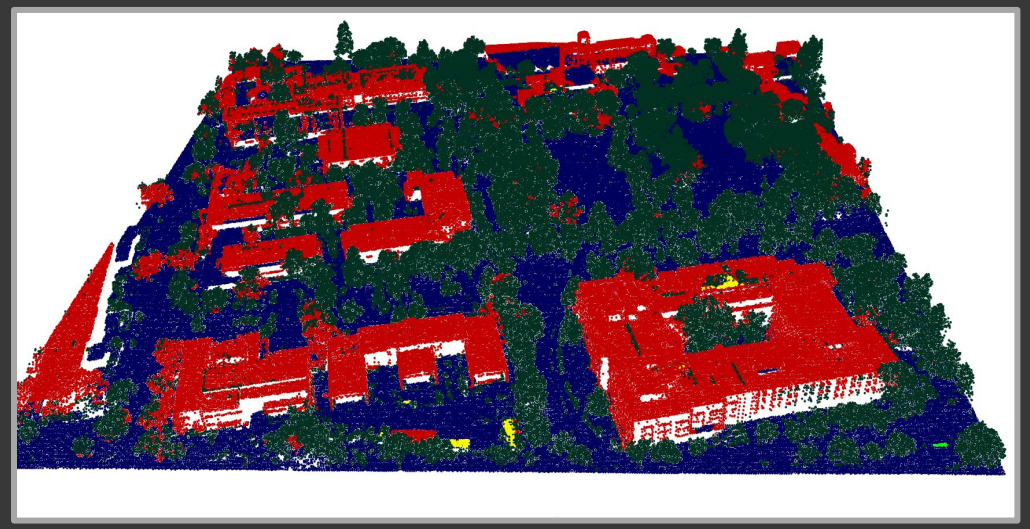
Machine type	n1-standard-16
CPU platform	Intel Broadwell
Architecture	x86/64
vCPUs to core ratio	–
Custom visible cores	–
Display device	Disabled Enable to use screen capturing and recording tools
GPUs	1 x NVIDIA T4 Virtual Workstation



OUR APPROACH

Testing Model on USGS Data





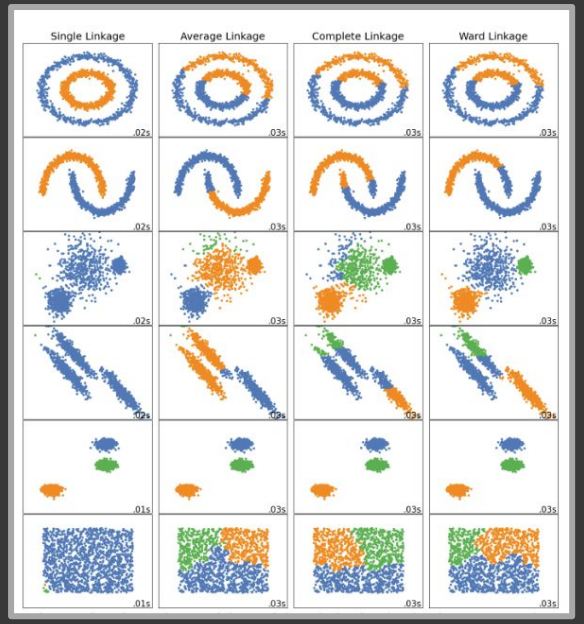
Our testing on USGS data resulted in buildings classified as ground in some cases, especially if the roof was flat

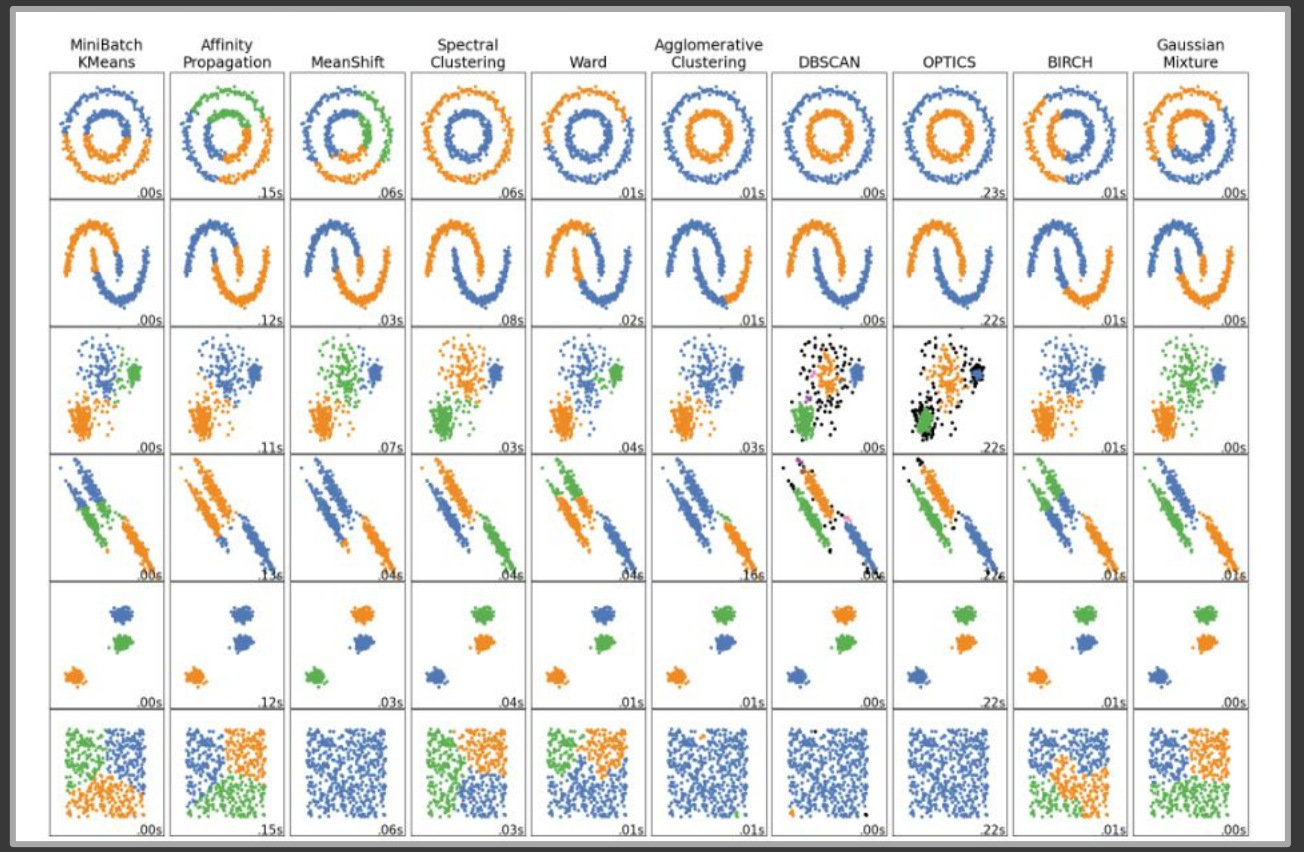
The solution: calculate the local ground average z-value (height above ground level) and reclassify the tile

OUR APPROACH

Segmentation & Clustering

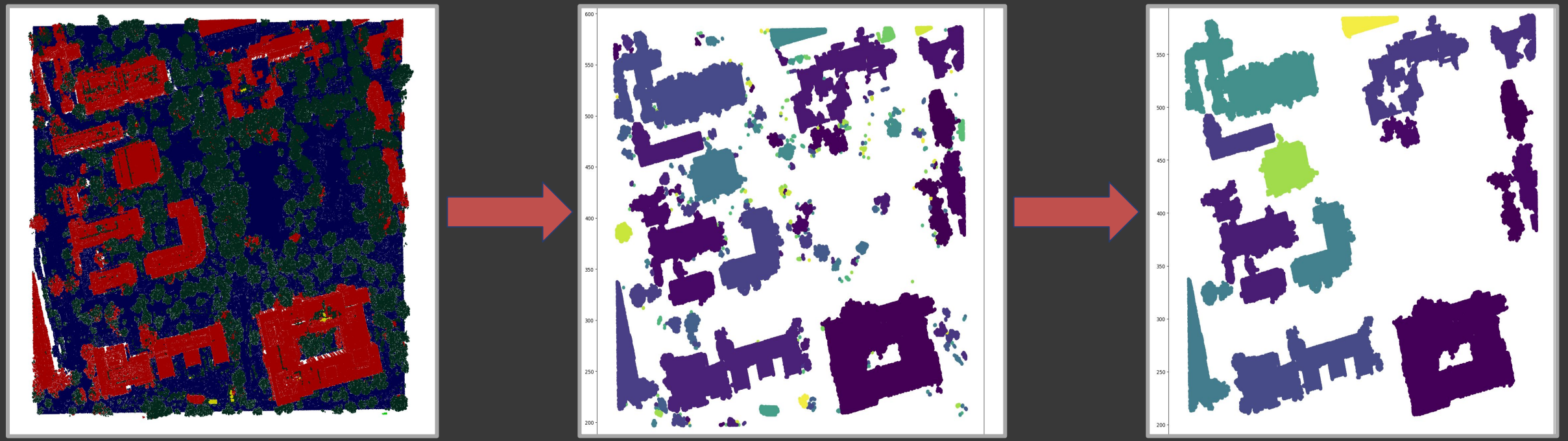
- Run clustering algorithm on points each class separately
- Multiple segmentation methods were explored
- Most of the clustering algorithms use the bottom up approach which are very memory intensive (requires at least 60GBs - 100GBs of RAM)
- Agglomerative Clustering with Single linkage using the distance threshold





OUR APPROACH

Clustering and Noise

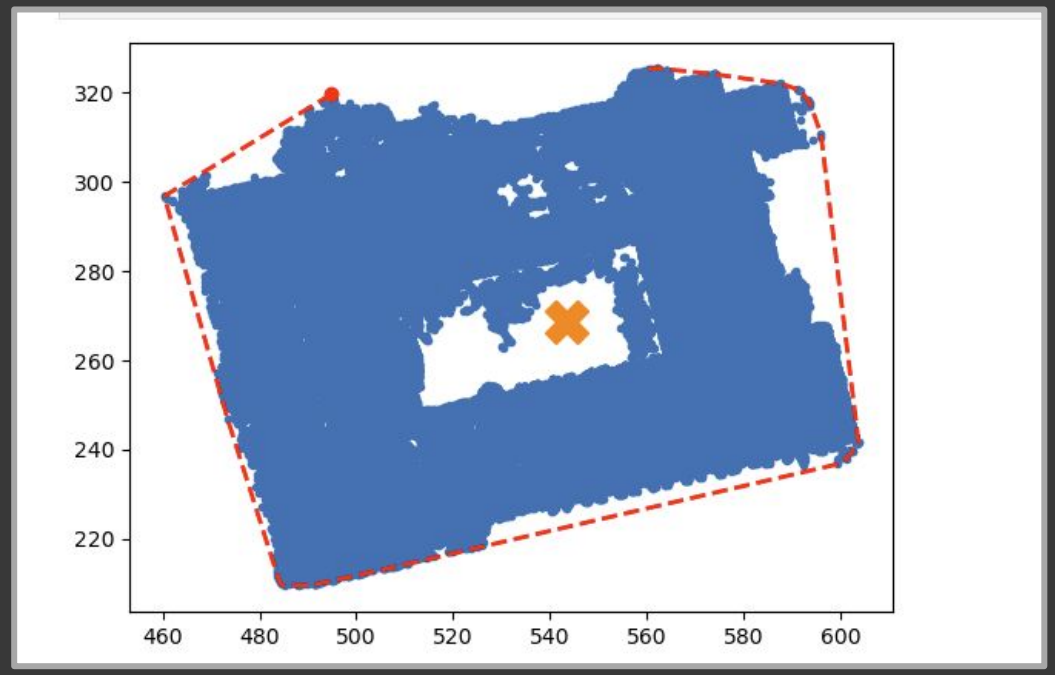


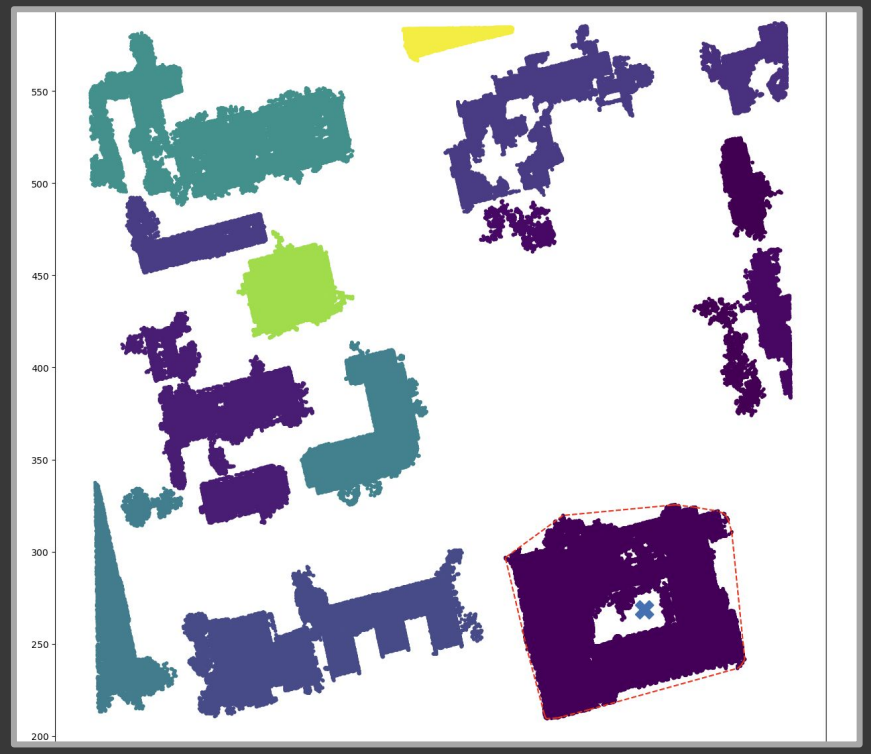
- We observed an overclassification of buildings. We knew we had to remove some of the noise
- To combat this overclassification, we used a custom function to remove clusters of point clouds made up of less than 1000 points
- In this example, initial clustering returned 312 clusters. After removing the noise, we ended up with 16 clusters

OUR APPROACH

Convex Hull

- A convex hull bounds the clusters with a bounding box, using the set of it's extreme points
- We calculated the 'centroid' of the bounding box using the mean of all x-values and y-values
- This point was then used as a pinpoint for the object, which provides a faster database query, compared to using the vertices of the bounding box





OUR APPROACH

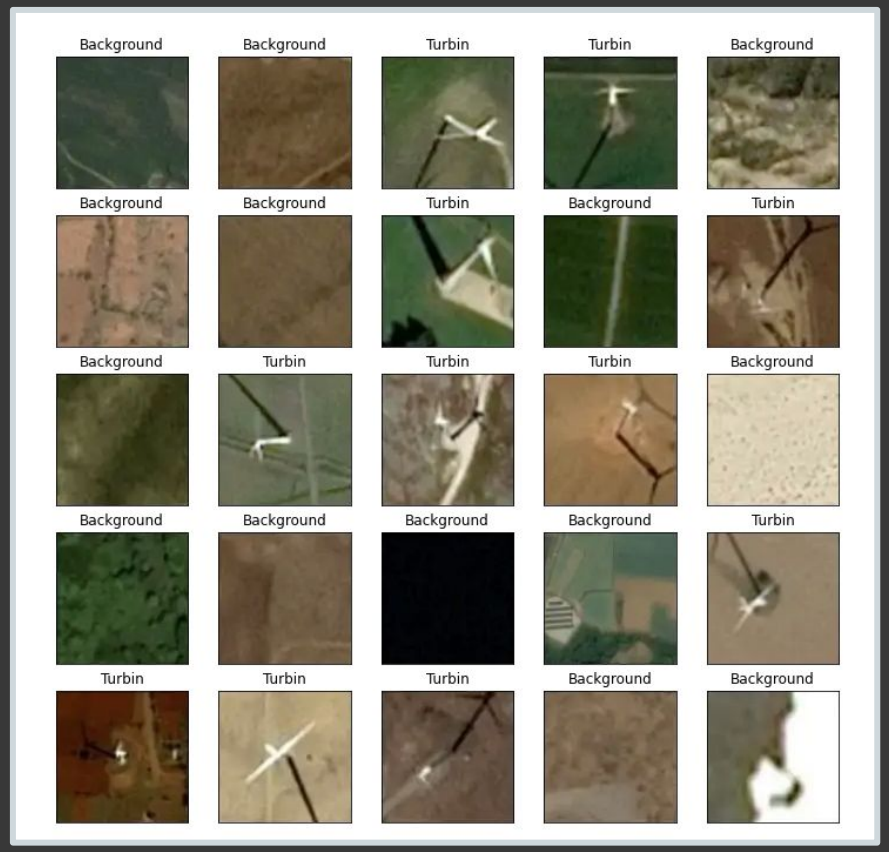
Our first model is successful for classifying static objects. However, we needed a solution to identify dynamic objects

OUR APPROACH

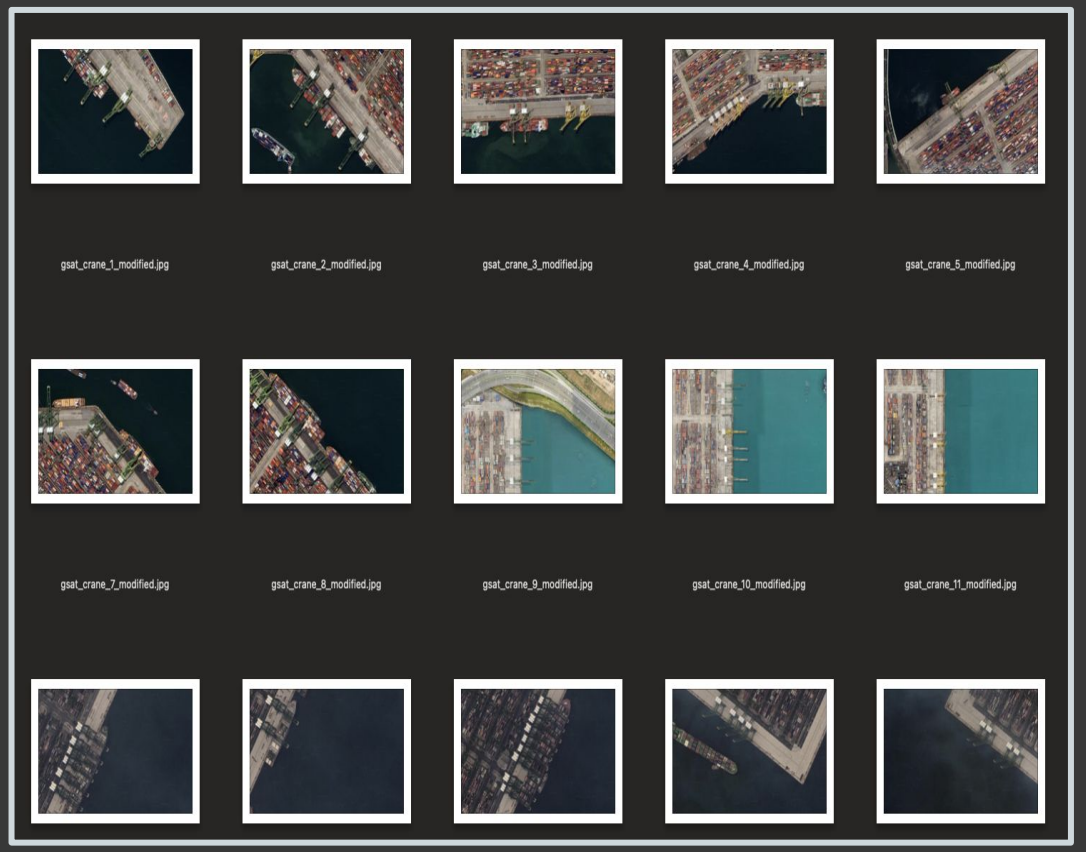
Classification

Satellite Image Classification to handle dynamic obstacles

Dataset: Airbus SPOT wind turbines patches



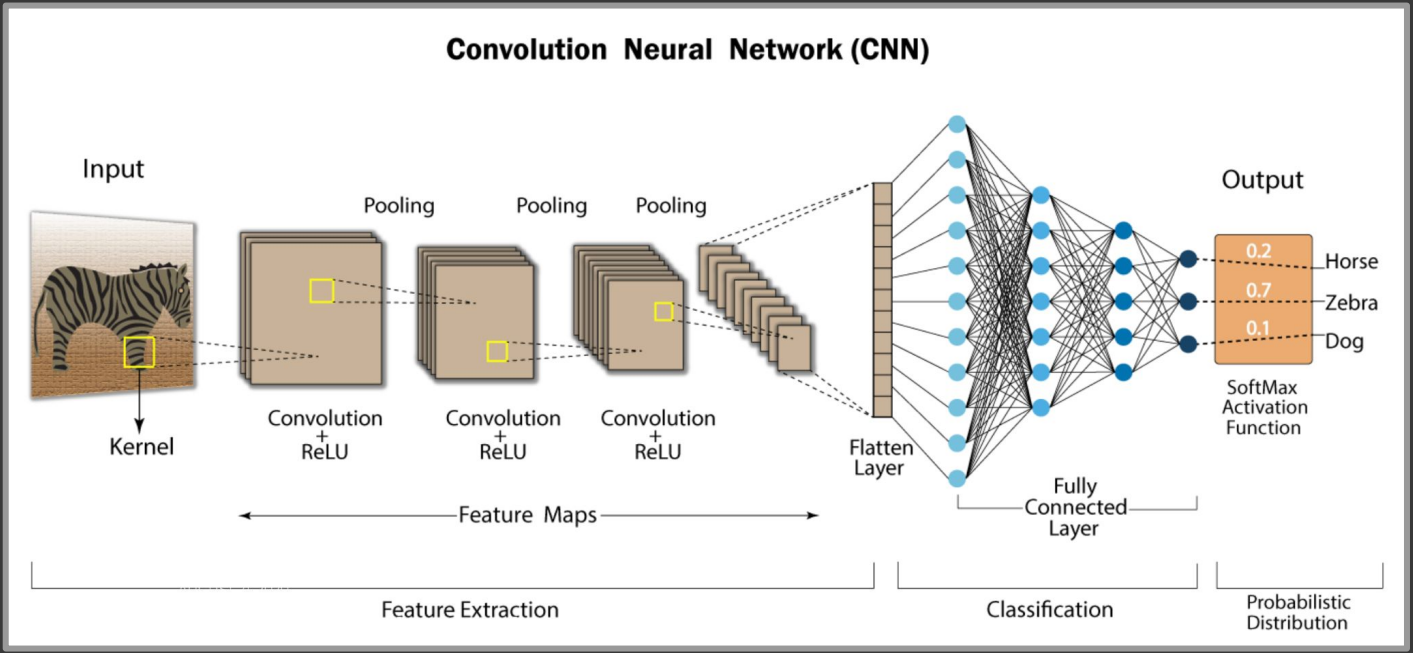
Dataset: IEEE DataPort 0.5-m resolution of cranes



OUR APPROACH

Second Model - Convolution Neural Network (CNN)

- Three Convolutional Layers (32, 32, 64 filters)
- 128 hidden neurons on dense/fully connected layer
- Total of 7.5 million parameters
- Transformation & Image Augmentation using Keras: (zoom, flip, rescale, shear etc.)
- Achieved 99.7% accuracy




```
cnn.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 254, 254, 32)      896

 max_pooling2d (MaxPooling2D  (None, 127, 127, 32)     0
 )

 conv2d_1 (Conv2D)           (None, 125, 125, 32)      9248

 max_pooling2d_1 (MaxPooling  (None, 62, 62, 32)       0
 2D)

 conv2d_2 (Conv2D)           (None, 60, 60, 64)        18496

 max_pooling2d_2 (MaxPooling  (None, 30, 30, 64)       0
 2D)

 flatten (Flatten)           (None, 57600)             0

 dense (Dense)               (None, 128)               7372928

 dense_1 (Dense)             (None, 1)                 129

=================================================================
Total params: 7,401,697
Trainable params: 7,401,697
Non-trainable params: 0
```

OUR APPROACH

Scalability Challenges

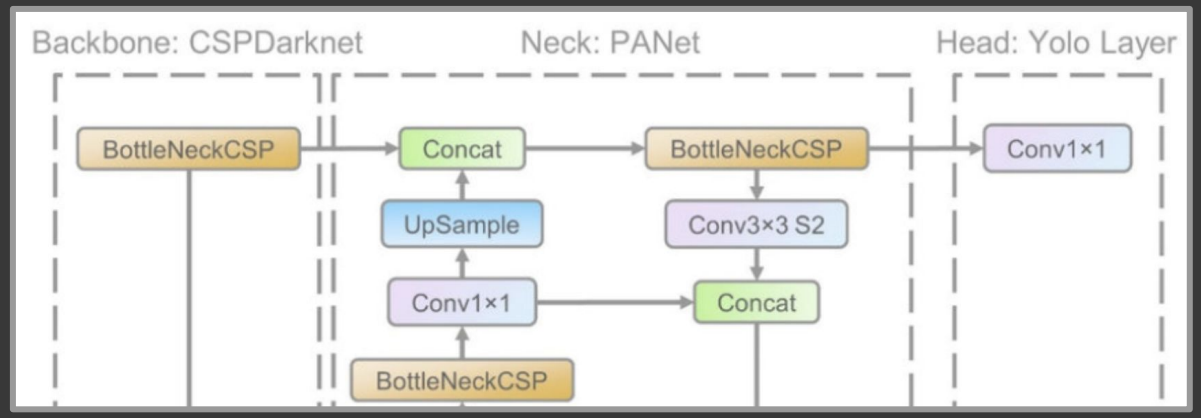
Challenges:

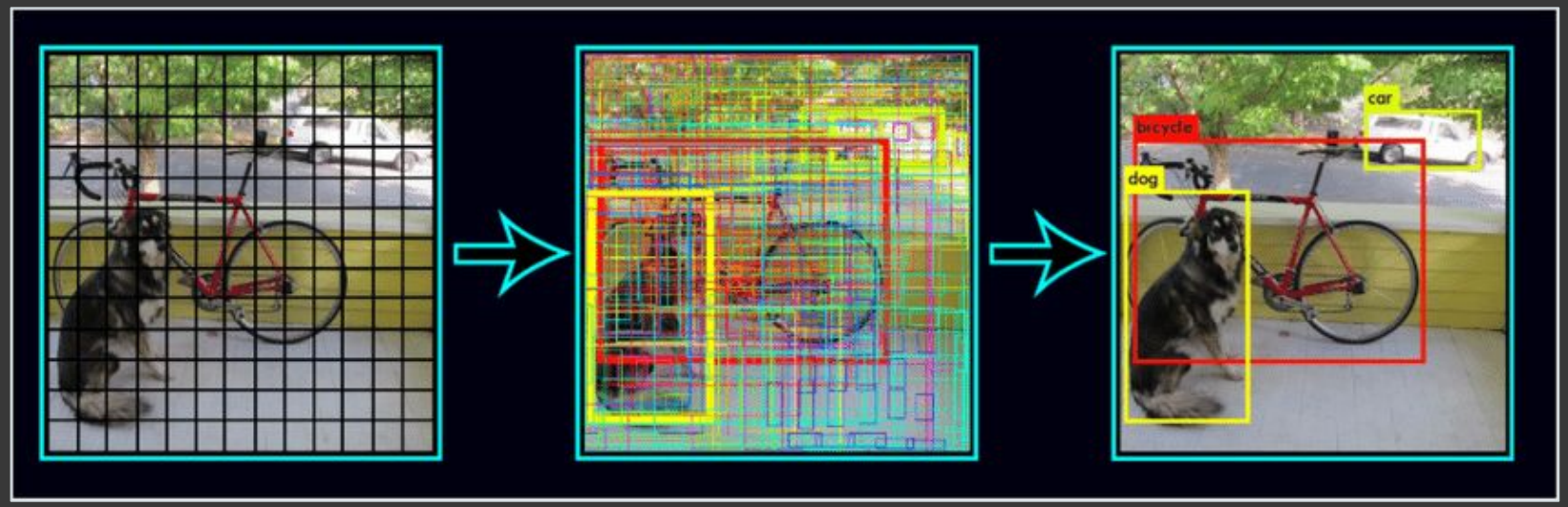
- Unable to handle multiple objects
- Unable to provide location
- Object must be centered within image
- Object must be zoomed in
- Computational and monetarily taxing to scale up

OUR APPROACH

Object Detection with Bounding Box: YOLO

- YOLO algorithm (You only look once) single stage detector
- Object detection algorithm that divides images into a grid system
- Each cell in the grid is responsible for detecting objects within itself





OUR APPROACH

Assemble the Dataset

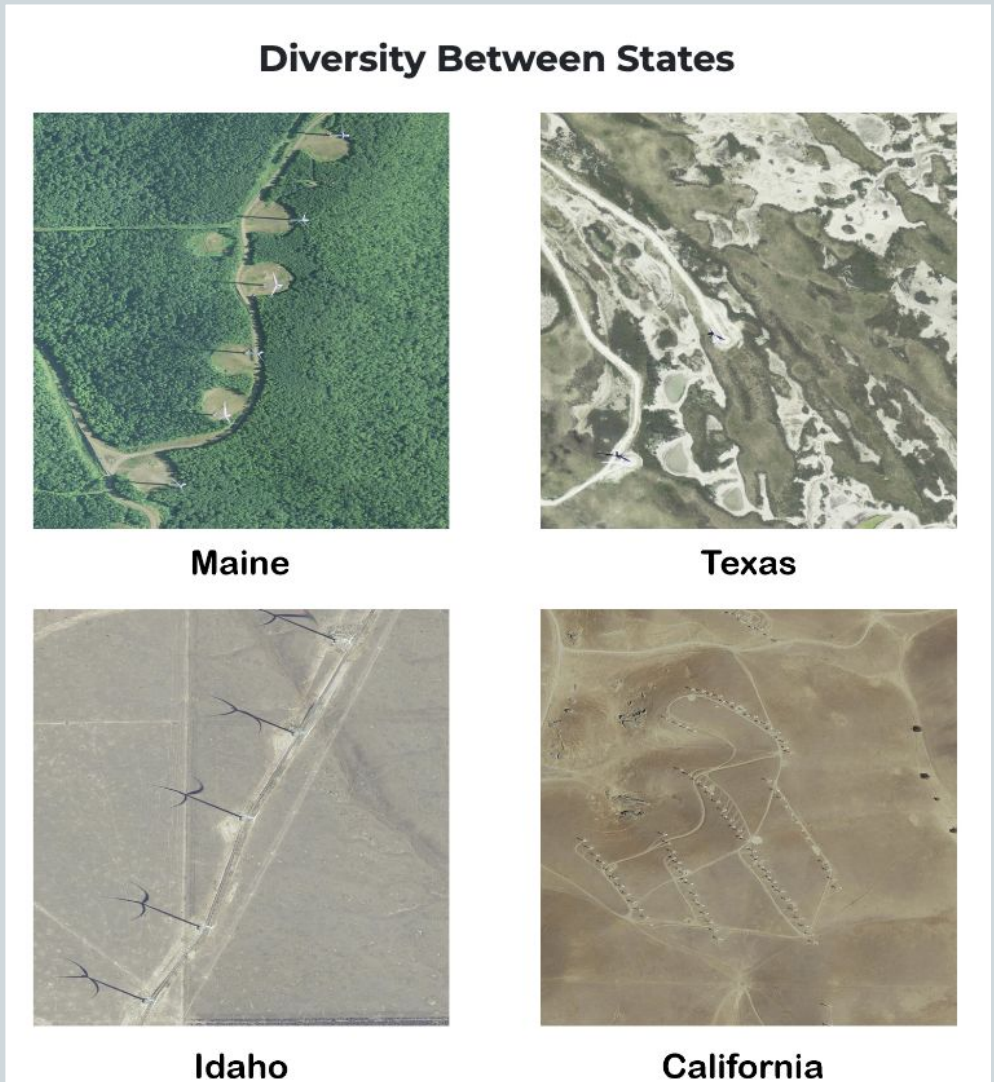


Figure: Images are taken from National Agricultural Imagery Program (NAIP) Power Plant Aerial Imagery Dataset.

- NAIP Power Plant Aerial Imagery Dataset
- 4454 Overhead images across the US from California, the Midwest and Texas
- Has variations in land types, wind turbine shapes and sizes across states
- By including variation in features like geography, wind turbine size, and shadow size in the training dataset, the model can generalize more broadly

OUR APPROACH

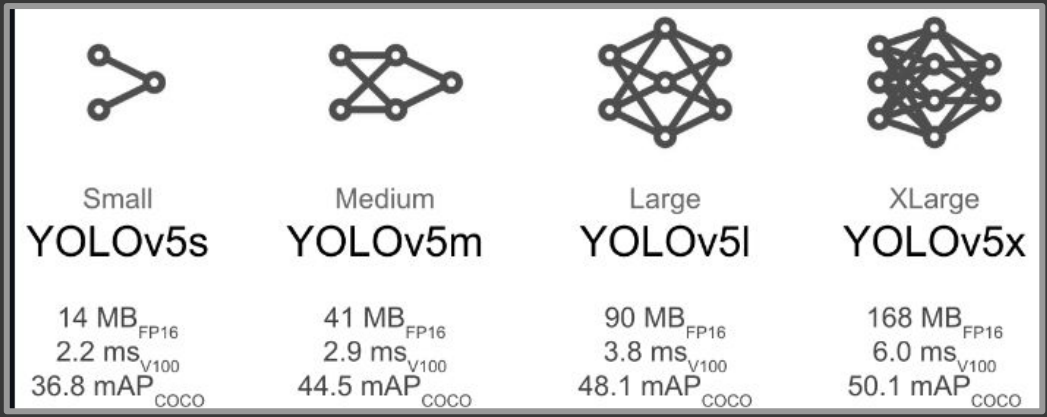
Roboflow

Using Roboflow:

- Convert an existing dataset to YOLOv5 format
- Upload raw images and annotate them in Roboflow with Roboflow Annotate
- About 800 images were annotated for model training and validation
- Augmentations (Resize, zoom, flip and rotation) were applied

OUR APPROACH

Model Training



- We used a Pytorch implementation of YOLOv5 Model to make use of the Roboflow dataset
- We chose the generic YOLOV5x (XLarge) checkpoint pretrained on Common Object in Context Dataset (COCO)
- 150 Epochs
- 322 layers
- >8 million parameters

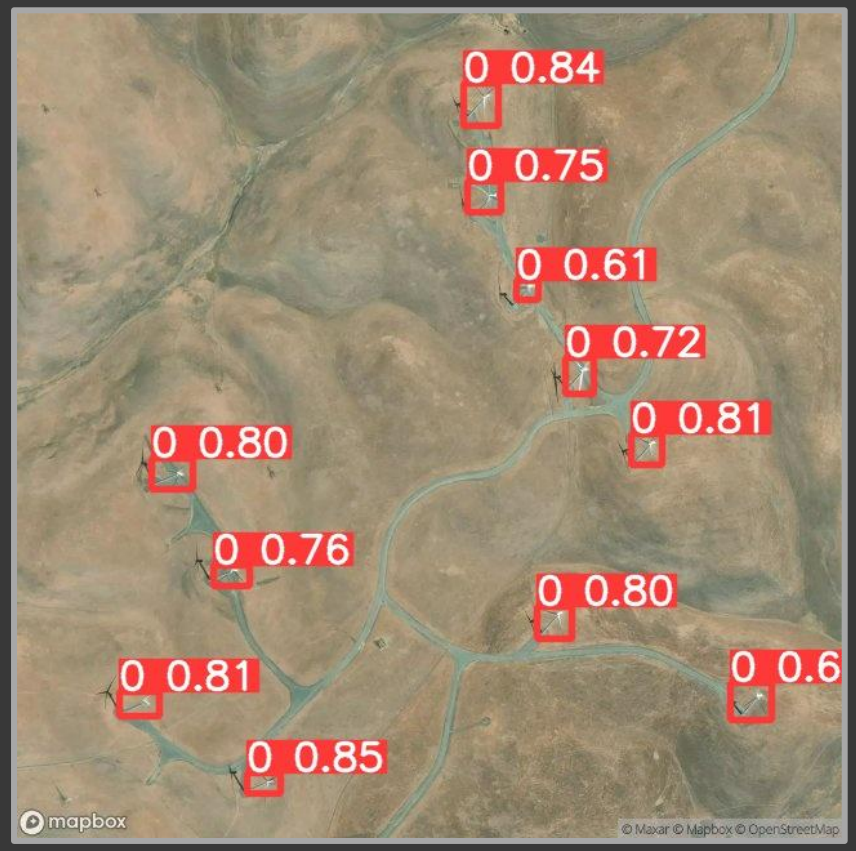
```
150 epochs completed in 2.374 hours.
Optimizer stripped from runs/train/exp3/weights/last.pt, 173.1MB
Optimizer stripped from runs/train/exp3/weights/best.pt, 173.1MB

Validating runs/train/exp3/weights/best.pt...
Fusing layers...
Model summary: 322 layers, 86173414 parameters, 0 gradients, 203.8 GFLOPs
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 3/3 [00:02<00:00,  1.04it/s]
                   all         70        373      0.731      0.708      0.685       0.25
Results saved to runs/train/exp3
```

OUR APPROACH

Model Results

- Tested in a real world dynamic setting
- Near perfect accuracy
- Baseline parameters were reduced due to safety concerns

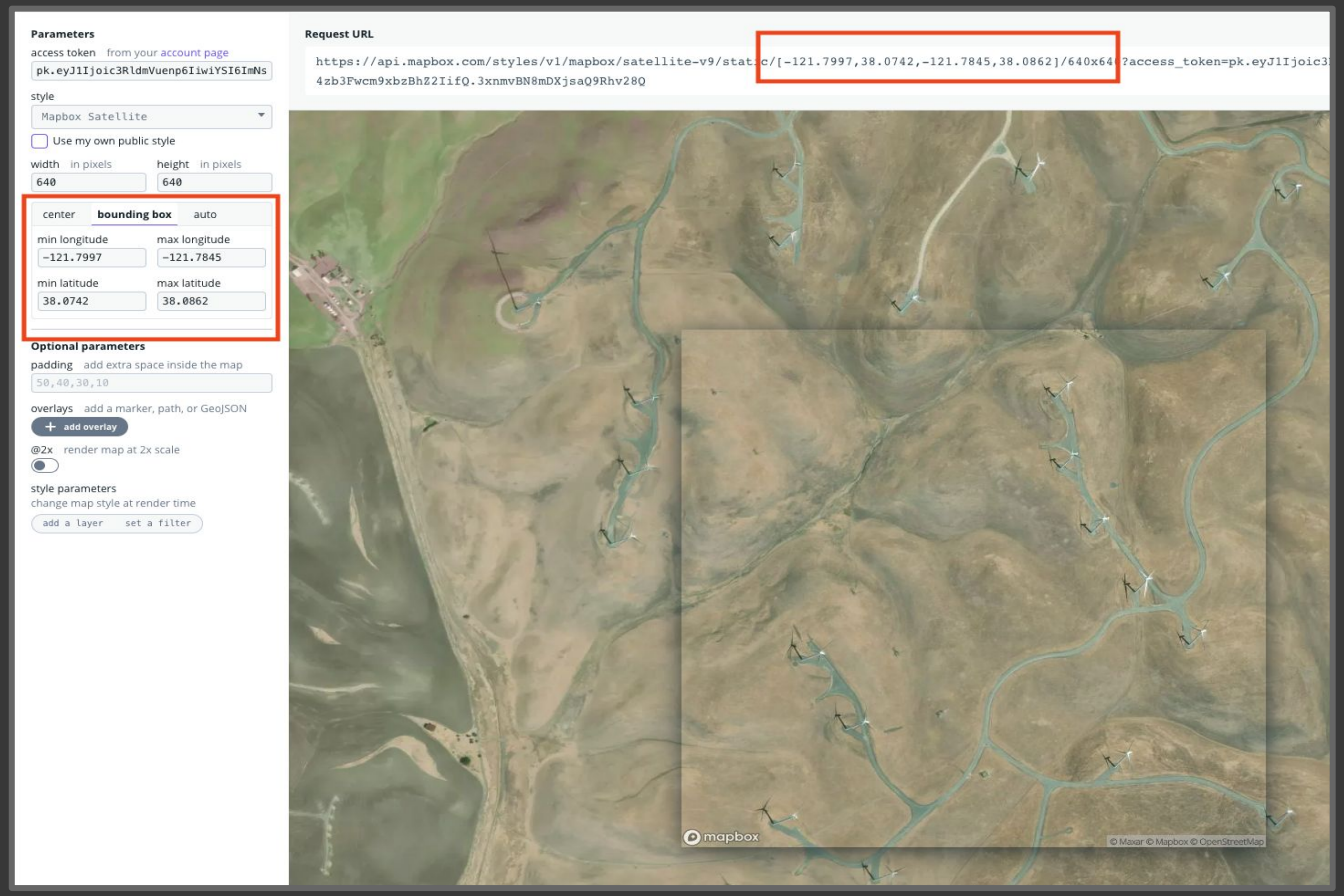


OUR APPROACH

Bounding Box to Geo Location

zidane.txt

```
0  0.481719 0.634028 0.690625 0.713278
0  0.741094 0.524306 0.314750 0.933389
27 0.364844 0.795833 0.078125 0.400000
```



Each row → Class, class x_center, y_center, width, height

OUR RESULTS

SQL Database

	Object	HAGL	Center	Vertices
0	Building(s)	10.79	[38.542552457073164,-121.75018286955013]	[38.542589003007436, -121.75059566322228],[38....
1	Building(s)	28.67	[38.54224087158561,-121.74948327765163]	[38.54198864246644, -121.75031621258651],[38.5...
2	Building(s)	16.89	[38.542405173389525,-121.74812607876285]	[38.54250342930847, -121.7485321144085],[38.54...
3	Building(s)	31.25	[38.542138962823564,-121.75192774428984]	[38.542291635885064, -121.75102776608105],[38....
4	Building(s)	13.10	[38.541877287834126,-121.74831633755196]	[38.541767534024586, -121.74800868018991],[38....
5	Building(s)	18.87	[38.541149374871544,-121.74806758924065]	[38.54138842665915, -121.74826240589809],[38.5...
6	Building(s)	20.21	[38.54098145020309,-121.74838440926165]	[38.54070194020875, -121.74820488813141],[38.5...
7	Building(s)	21.11	[38.539740390475984,-121.74898894191779]	[38.54020549320194, -121.74953724169754],[38.5...
8	Building(s)	29.48	[38.54162312730218,-121.75213536398749]	[38.541654393264814, -121.75251207130829],[38....
9	Building(s)	16.42	[38.54162414107391,-121.74978670225218]	[38.54170032339375, -121.75004249482232],[38.5...
10	Building(s)	12.96	[38.541345884443864,-121.75138283328235]	[38.54161502949612, -121.75150772970817],[38.5...
11	Building(s)	16.70	[38.54073896608736,-121.75189648280059]	[38.54019055891056, -121.75159133412446],[38.5...
12	Building(s)	15.50	[38.540588390193356,-121.75086420944267]	[38.54049801890671, -121.75132796884193],[38.5...
13	Building(s)	17.10	[38.53971263266667,-121.75262723815835]	[38.539410481866135, -121.75223897733719],[38....
14	Building(s)	15.55	[38.54027602574538,-121.75238704727289]	[38.540281701972944, -121.75256687983682],[38....
15	Building(s)	20.60	[38.53959957908259,-121.75125555976568]	[38.53964815933765, -121.75010566891191],[38.5...

Object:
- Classified obstacle

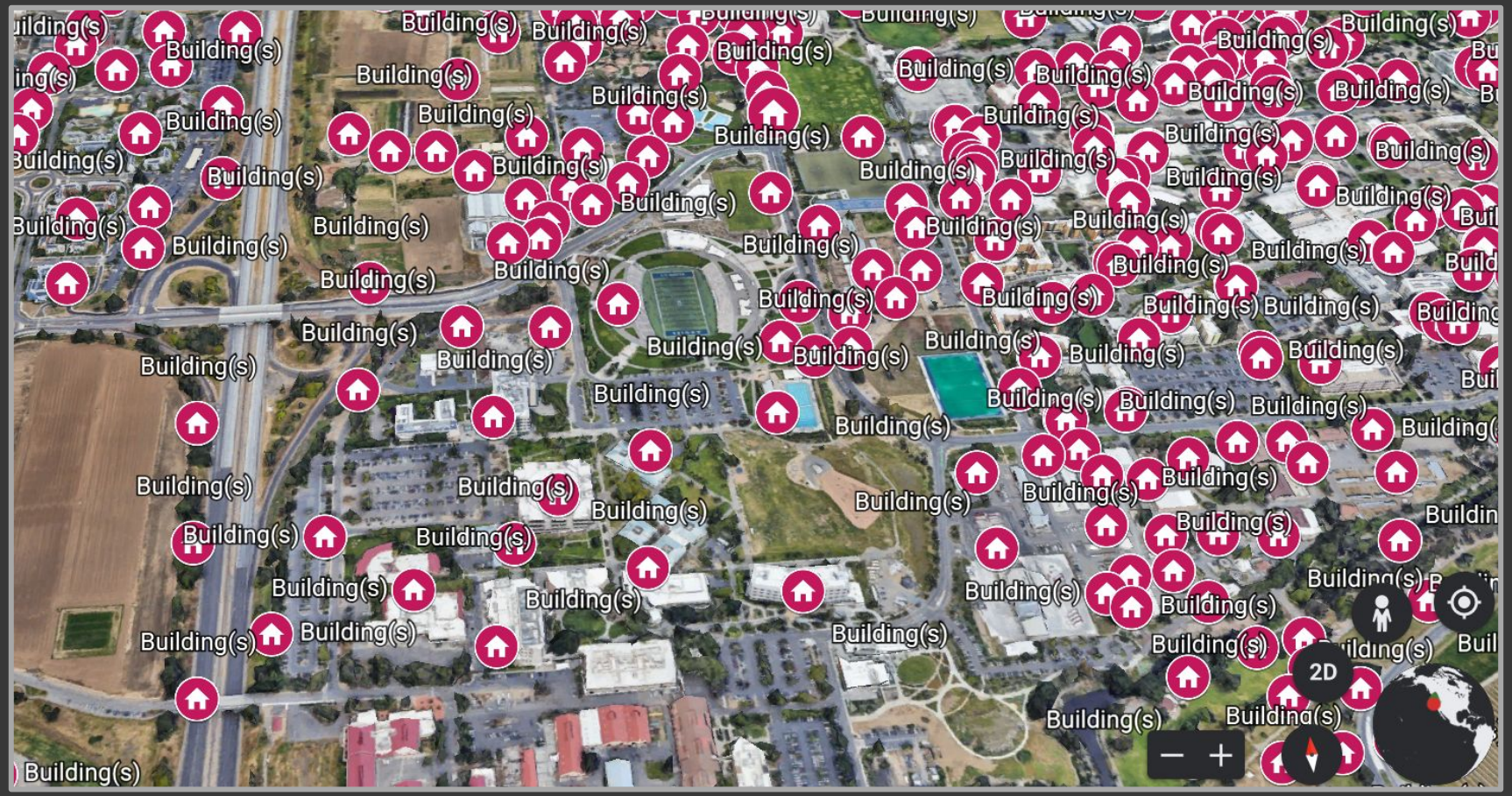
HAGL (Height Above Ground Level):
- Calculated (static)
- Assigned fixed value (dynamic)

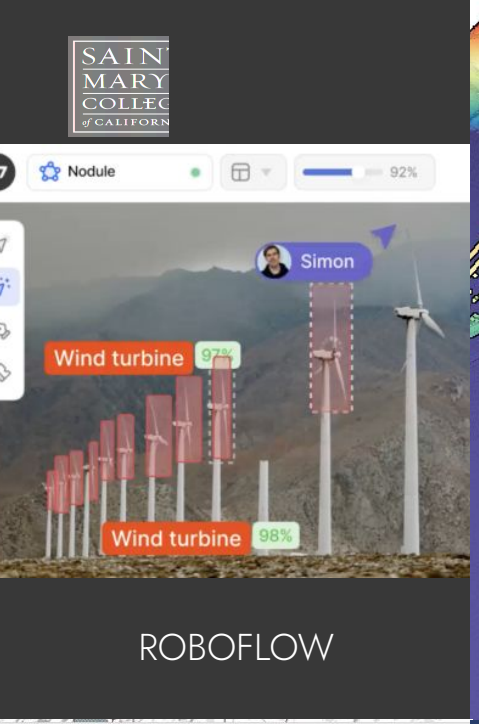
Center:
- Centroid of convex hull (static)
- Width and height relative to image (dynamic)

Vertices:
- Outer extreme points of convex hull

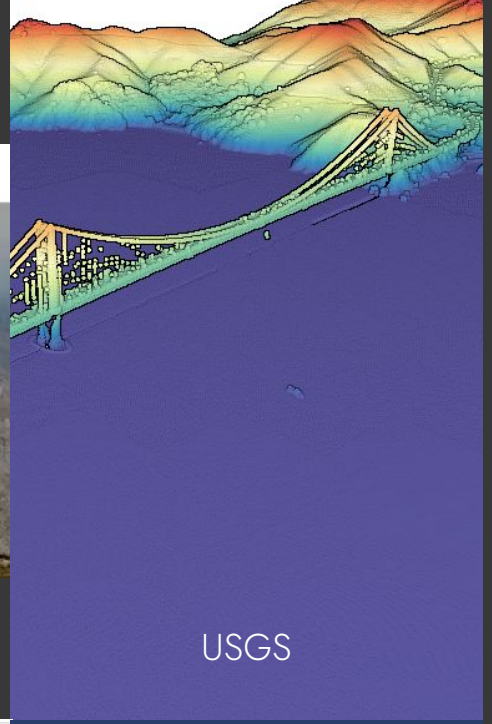
Live Demo

Link

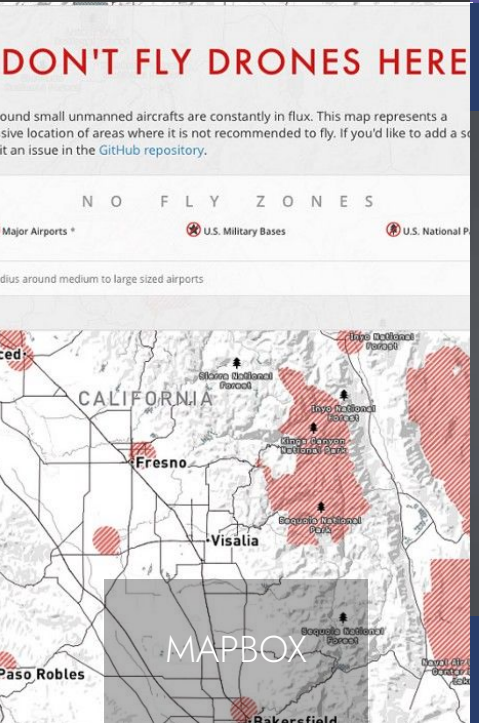




ROBOFLOW



USGS



MAPBOX

GitHub

Tech Stack

Tools

Tools

Roboflow (Annotation/labeling Images)
TensorBoard(performance metrics)
MapBox API (Satellite images)
USGS API (Lidar data)
GCPVM (Virtual machine),
Ubuntu Desktop(Linux GUI),
Chrome Remote Desktop (VM remote access),
Tmux (Terminal Multiplexer),
Github (Collaboration & version Control),
JIRA (agile development & Project management),
Confluence (Documentations),

Tech Stack

Libraries

Framework

- Tensorflow
- Pytorch

Libraries

- Keras (Deeplearning API)
- trimesh (Dealing with meshes from binary/ASCII)
- plyfile (Reading and writing ASCII and binary PLY file)
- laspy (Lidar pointCloud io)
- utm (Bidirectional UTM-WGS84 converter)
- open3d (Data processing & visualization)
- sqlite3 (Interaction with SQL database)
- Common Lib
- Pandas, matplotlib, sklearn, sys, os, cv2, numpy, PIL, traceback, scipy, glob

Database & DBMS:

- SQL & mySQL

FUTURE WORK

What we still want to explore

CI/CD pipeline (constant updates of the Database)

Faster and more specialized database in geospatial data: PostGIS, Snowflake, Redis

Structure from motion (another object classification method)

Optimization (using classified objects to determine risk and path)

Improve our models by training on more annotated data (more ‘good’ data is more important than better algorithms)

Explore a more efficient framework for point cloud classification (downsampling & TONIC)

Thank You

Q & A