| | | | |
|---|---|---|---|
| **Dataset/** | **Region 1/** | Cloud Coverage Low/ | Images + Labels |
| | | Cloud Coverage Mid/ | Images + Labels |
| | | Cloud Coverage High/ | Images + Labels |
| | | Horizon / | Images + Labels |
| | **Region 2/** | Cloud Coverage Low/ | Images + Labels |
| | | Cloud Coverage Mid/ | Images + Labels |
| | | Cloud Coverage High/ | Images + Labels |
| | | Horizon / | Images + Labels |
| | **...** | ... | ... |
| | **Region N/** | Cloud Coverage Low/ | Images + Labels |
| | | Cloud Coverage Mid/ | Images + Labels |
| | | Cloud Coverage High/ | Images + Labels |
| | | Horizon / | Images + Labels |

**Blockers:**

- **N/A**

**Requirements:**

- **N/A**

**Last week:**

- **Dataset:**
  - Finalized dataset requirements and basic structure of the dataset
- **Jetson Orin:**
  - Flashed machine image and set up SSH connection
  - Ready to deploy model
- **Pipeline:**
  - Add visualization demo
  - Improve models

**Next step**:

- Finalize the API tool for dataset building pipeline
- Run Inference on Jetson Orin

## Interfaces

Avionics:

- Determine power usage of Orin, coordinate this usage with power budget

GNC:

- Discuss about partitioning Orin CPU/GPU usage
- Determine GNC estimation software that needs to be run on Orin/how that interacts with duty cycling the Orin

# Tentative Dataset Requirements

| | | | |
|---|---|---|---|
| **Dataset/** | **Region 1/** | Cloud Coverage Low/ | Images + Labels |
| | | Cloud Coverage Mid/ | Images + Labels |
| | | Cloud Coverage High/ | Images + Labels |
| | | Horizon / | Images + Labels |
| | **Region 2/** | Cloud Coverage Low/ | Images + Labels |
| | | Cloud Coverage Mid/ | Images + Labels |
| | | Cloud Coverage High/ | Images + Labels |
| | | Horizon / | Images + Labels |
| | **...** | ... | ... |
| | **Region N/** | Cloud Coverage Low/ | Images + Labels |
| | | Cloud Coverage Mid/ | Images + Labels |
| | | Cloud Coverage High/ | Images + Labels |
| | | Horizon / | Images + Labels |

Regions: **~574**

- Each consists of 25 non-repeated raw LandSat images -> ~ 50 Training images
- Excluding regions contain only the ocean

Data Source:
**LandSat 8 Collection 2, Level 1 Products**
usgs.gov/landsat-missions/landsat-collection-2-level-1-data

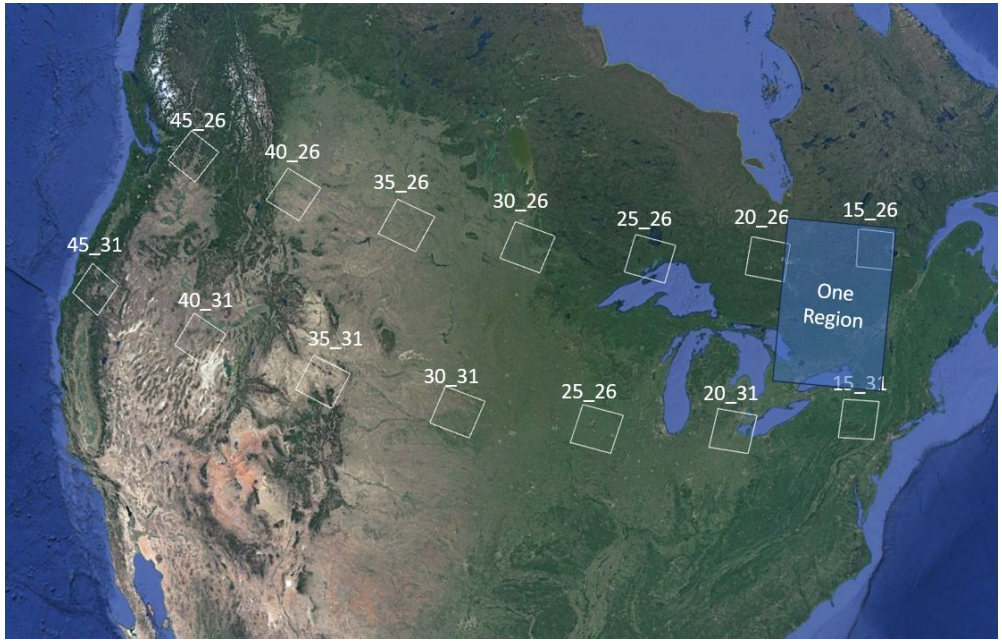
- Minimal Processing: include atmospheric distortions
- Improved Data Quality: Collection 2 offers enhanced absolute geolocation accuracy, updated digital elevation models, and better radiometric calibration

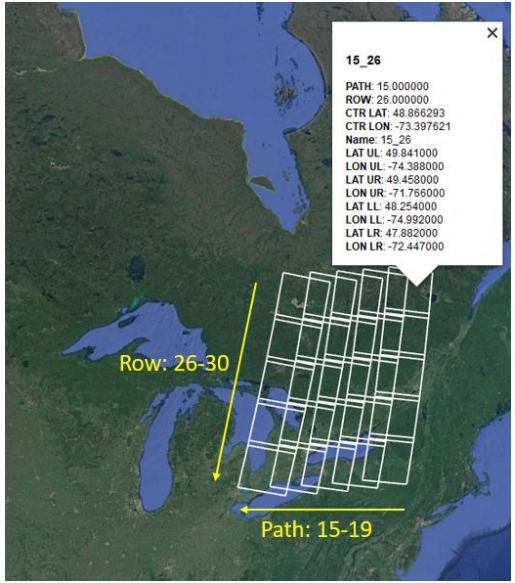
Date Range: **2021-2023**

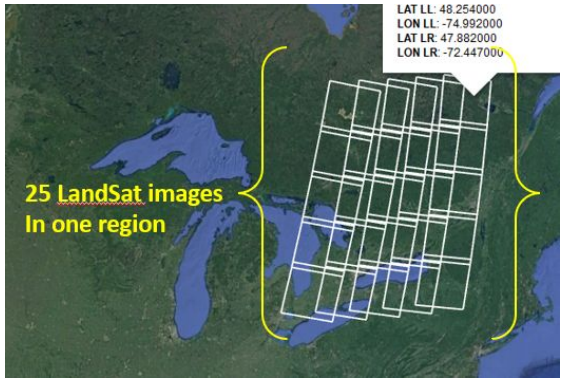
Training Image:

- Each Training image should have ~ 4 augmentations: Brightness, Contrast, Rotation

# Just a bit more context…



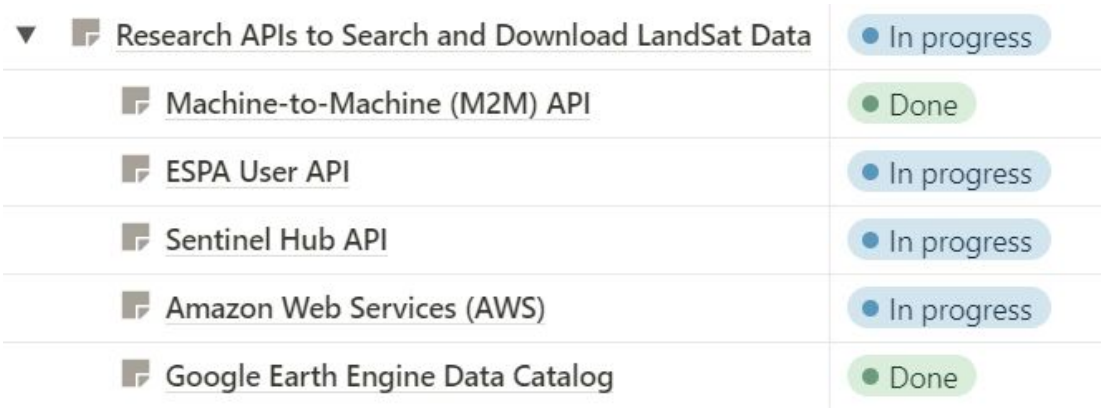



# Dataset Building Pipeline

Small pipeline building for popular LandSat access APIs

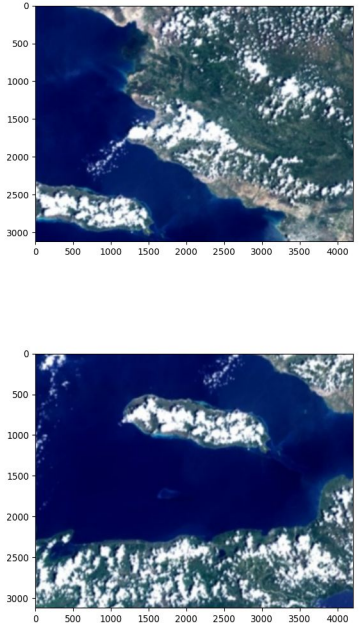
- Attempt to search and download images needed from one region and compare the data availability, time-efficiency, and cost-efficiency




Tutorial to batch download LandSat images from EarthExplorer:
https://riverflame-dev.notion.site/Download-Data-from-Earthrexplorer-6d60602d8797416f8d6c0d93e0611d49?pvs=4

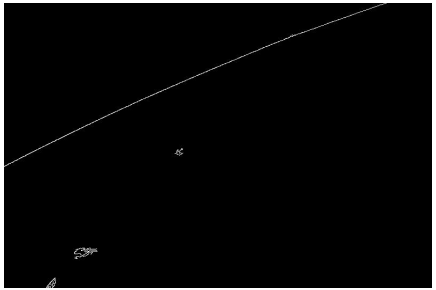
# Extract Camera Images from Dataset Images

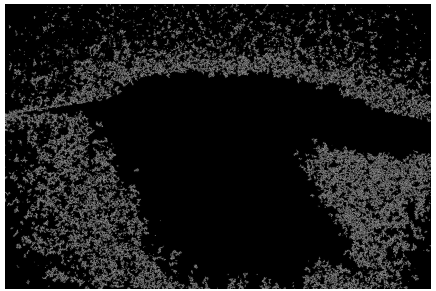
- Pick only top right corner due to the the overlapping paths and rows
- Same resolution & same swath as our physical camera



# Horizon Images

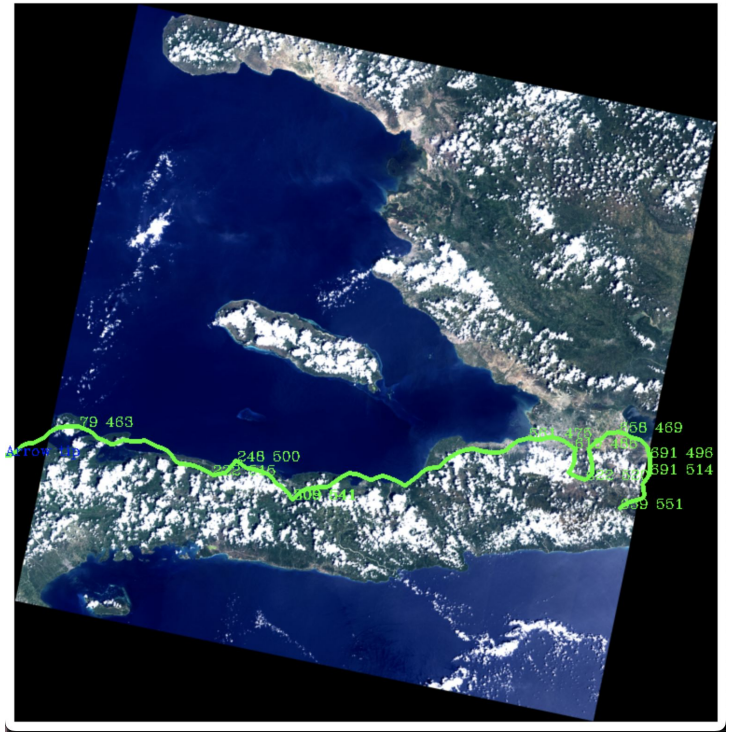
- Offers a consistent reference point for determining the satellite's orientation relative to Earth.
- The visible curvature can assist in estimating altitude, especially when coupled with a known field of view.
- Features of horizon images can improve model generalization
- Detect the edge between space & earth - not ideal results





# Landmark detection

- Using classical OpenCV features such as saliency and static saliency detectors, we are able to detect areas of interest on an image
- Potential avenues:
  - Ease and help automatic labeling for our training set by identifying landmarks and offsetting by known coordinates
  - Narrow down our dataset for inference, could potentially utilize smaller sub-models



**Pipeline:**
**Stage 1 - Region Classifier**
**Stage 2 -**

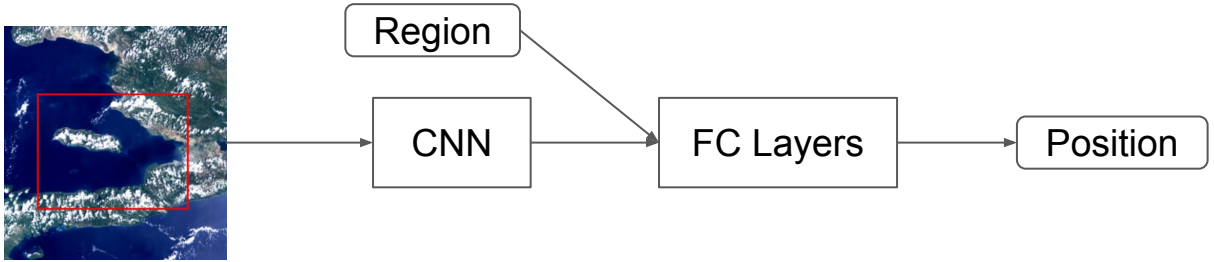
1. Landmark Detection
    a. Detect landmarks -> Given landmarks' positions -> Calculate our position

Region "A" → YOLOv8 A → Landmark → Position

[{ "**class**": "**island A**", "**confidence**": 0.98,
"**bounding_box**": {"x": 120, "y": 150,
"width": 50, "height": 75}  }]

**Pipeline:**
**Stage 1 - Region Classifier**
**Stage 2 -**

1. Landmark Detection
    a. Detect landmarks -> Given landmarks' positions -> Calculate our position
2. End-to-end:
    a. Image + Region -> Neural Network -> Our position




**Pros & Cons for Landmark Detection:**

1. more accurate if we can precisely locate the landmarks
2. more obvious and makes more "human" sense
3. may not be able to find landmarks in certain pictures
4. need lots of manually-labelled pictures & landmark data (position etc.)
5. more overhead in terms of post-processing network output