Alouette-1 – Ionogram Data Extraction Methodology

I. An Evergreen Document

The Canadian Space Agency (CSA) has developed an approach to extract data from scanned images of Alouette-1 ionograms, which were originally recorded on 35 mm negative films. However, the data extraction methodology presented is continually improving, and is being applied more generally to Alouette-2, ISIS-1 and ISIS-2 images of scanned ionograms. Therefore, it is expected that this document will be updated as new improvements are implemented.

This document will present in detail some of the errors in data processing and extraction. And it will provide instructions on how to re-process the data, so that one may take the approach presented and try to improve upon it.

It is helpful to gain an overview of the history of Alouette-1's data, and the data restoration efforts that occurred, which has led to the current set of Alouette/ISIS (Alouette-1, Alouette-2, ISIS-1, ISIS-2) scanned images:

- **Alouette-1 – A History of The Data from Canada's First Satellite Over 60 Years (2023) – Ravendra Naidoo**

There is more detailed information about the Alouette-1 scanned images and how to read them, which was written when scanning of the 35 mm film rolls started in 2017. Please be advised that the document below assumes the reference year of 1960 for the metadata analysis. For more information refer to section IV. Determining the reference year:

- **Alouette-1 Data Restoration (2017) – David Lessard**

Furthermore, much work was done from 2017 – 2022 that are included in the archive.

II. Data Acquisition

The images (…/Alouette-1/data/) are organized into Directories and Subdirectories. Directories correspond to a box of 35 mm negative film rolls; and Directories tend to start with the letter 'R'. Then, a particular Subdirectory within a Directory corresponds to a particular 35 mm film roll. Within that Subdirectory is a set of images, which are the scanned ionograms on that film roll (see Figure 1).

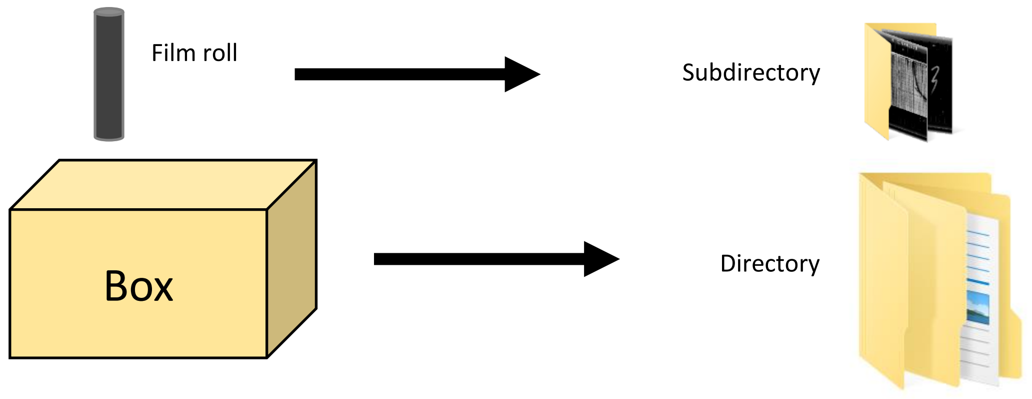


Figure 1: Subdirectories are folders of scanned images of ionograms from a film roll. A Directory corresponds to a box of film rolls.

2.1 FTP Entire Download

The simplest way to acquire the raw scanned images and the extracted ionograms is to download it entirely from the CSA FTP, using the following details:

> **FTP server:** data.asc-csa.gc.ca
> **Username:** anonymous
> **Password:** [users must enter their email address]
> **Path to Alouette-1 data:** users/OpenData_DonneesOuvertes/pub/Alouette-ISIS/Alouette-1/

Then in the directory for your project, rename the three folders:

- data → 02_downloaded
- result → 05_result
- log → 06_log

2.2 FTP Download by Subdirectories

However, it is an extremely large amount of data approximating **~ 2.6 TB**. An alternative approach was developed to download subdirectories of data one-at-a-time.

To download subdirectories one-at-a-time, first run the following Jupyter notebook:

- **00_Download Subdirectories from FTP.ipynb**

Follow the instructions in the notebook to set-up folders correctly in your project directory. Then test downloading of subdirectories.

Once subdirectories are downloading correctly, the process can be continued on a loop, by running the following Python script from the command line with 3 arguments:

```
Alouette_downloader.py [the file path to the root directory for your
project] [user's email address for FTP connection] [username for
download log]
```

For example:

```
python Alouette_downloader.py "C:/Projects/Alouette-1/data/"
"x@xmail.co" "UserX"
```

For the program to work, you will need to download the 'log' folder from the FTP directly and save it in the root directory for your project. Then rename the 'log' folder to '06_log'. Please note that while the decision of renaming the folder is entirely optional, the code is written based on this assumption and a need for changing the name of the folders on the code will rise if a different method is followed.

III. Inspection of Extracted Ionograms

Once you have acquired data, it is a good idea to inspect a sample of ionograms along with its corresponding extracted ionogram points. Run this Jupyter notebook:

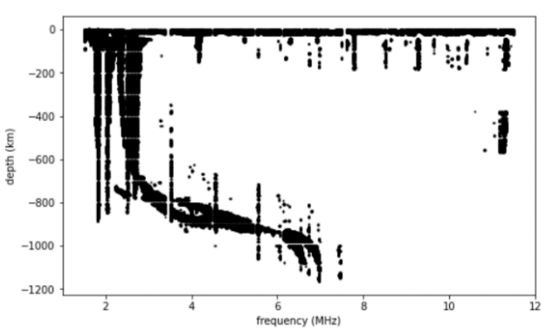
- **01_Inspect Extracted Ionogram Points.ipynb**

If you had only downloaded a few Subdirectories of images following section 2.2, you will also need to download the corresponding mapped coordinates. Go to the FTP source for Alouette-1 mapped coordinates:

- /users/OpenData_DonneesOuvertes/pub/Alouette-ISIS/Alouette-1/result/mapped_coords/

Download the corresponding mapped coordinates for the subdirectories you wish to inspect. Save these in '05_result/mapped_coords/' in your project root directory.

```
(55853, 2)
```



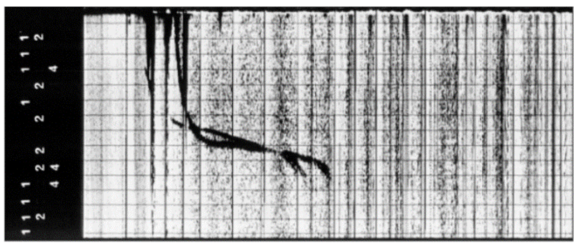


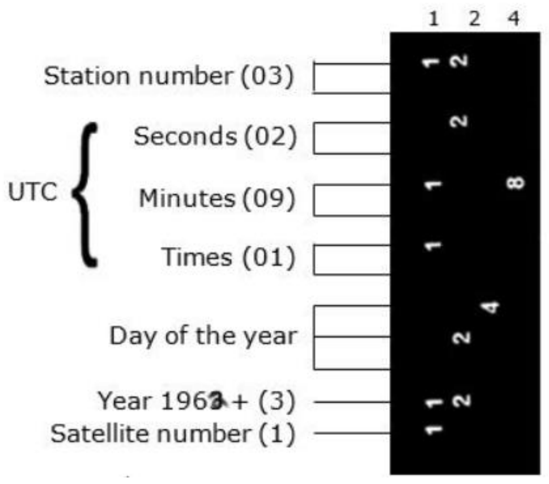
Figure 2: (left) Extracted datapoints mapped to a depth-frequency space; (right) the corresponding scanned image of the ionogram.

IV. Determining the reference year

Alouette-1 Ionograms has three types of metadata encoded in one of the three metadata types:

- 'dot',
- 'num'
- 'num2'

The 'dot' type has its metadata in the punctuated format. The 'num1' type has its metadata in the numerical format and is typically seen on the left side of the ionogram. An example of a num 1 type metadata could be seen on Figure 2, above. 'Num2' type is also numerical, yet the metadata is located on the bottom side of the ionograms.



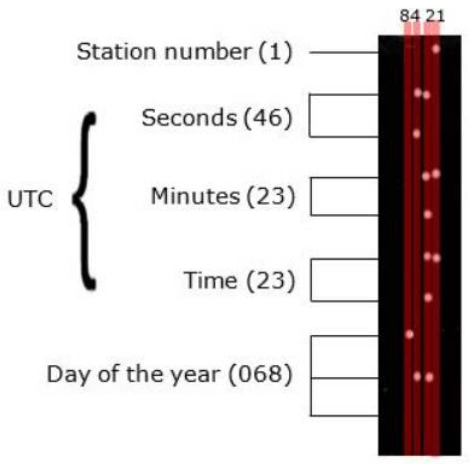


Figure 3: (Left) Example of the process of reading the metadata from the 'Num1' type; (Right) Example of the process of reading the metadata from the 'dot' type.

As shown on the left image of Figure 3, to identify the year encoded in the metadata for num1, the sum of the numbers shown on the ionogram is added to the base year of 1962. It is worth mentioning that in some documentations such as Alouette-1 Data Restoration (2017) by David Lessard the base year was identified as 1960. However, the baseline of 1962 was deemed more suitable for the following reasons:

1) Considering that the Alouette-1 satellite was launched in September of 1962 then by choosing the baseline of 1960 as shown on the left image in Figure 4, a portion of the ionograms would have been processed before the official launch of the satellite which the team deemed unlikely. Whereas, by taking the baseline as 1962, the graph shows a better distribution for the ionograms based on the year launched.

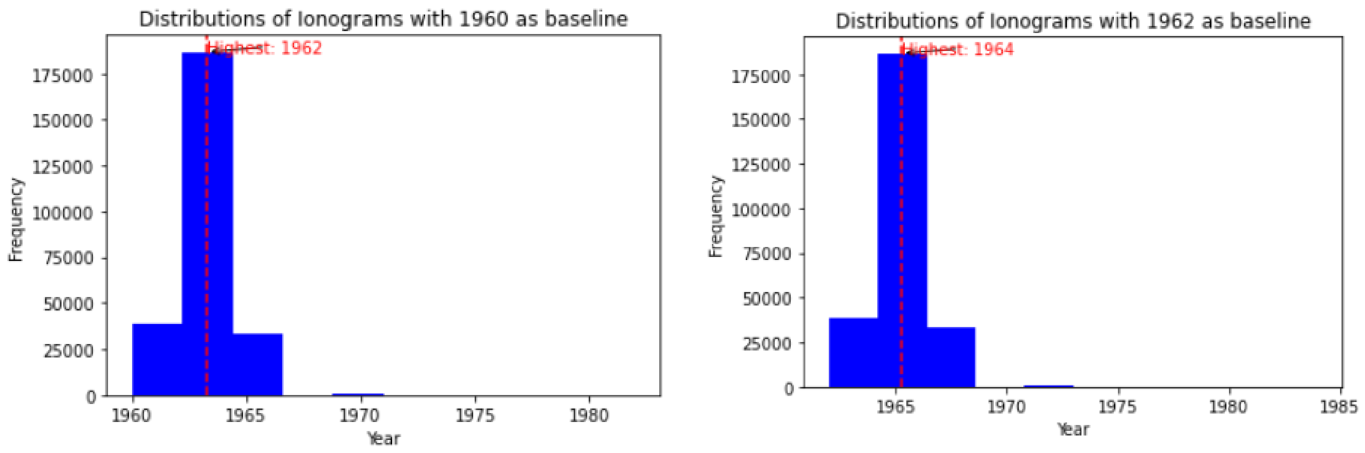


Figure 4: (Left) Frequency of the processed ionograms with the baseline of 1960; (Right) Frequency of the processed ionograms with the baseline of 1962.

2) The satellite tracking processing of Alouette demonstrated improved consistency with the station's location and the anticipated satellite observation time, aligning more closely with the 1962 baseline rather than the 1960 reference point.

The adjustment of the baseline was not necessary for the 'dot' and 'num2' metadata types. Figure 5 shows an example of reading the 'num2' metadata types:

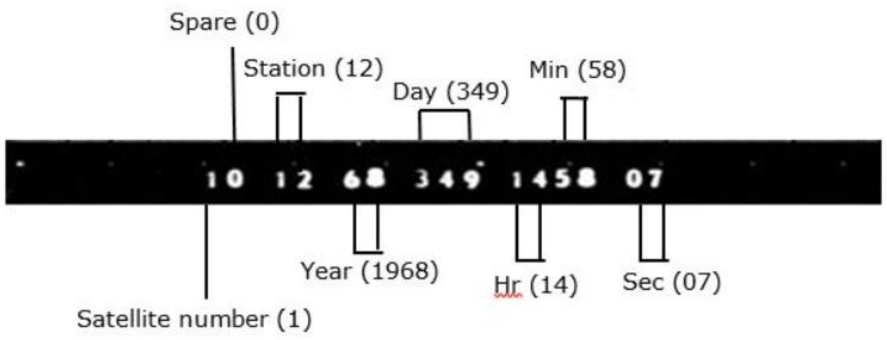


Figure 5: Example of the process of reading the metadata from the 'Num2' type

V. Working with the Processed Data

The CSA has already processed the entire set of all available Alouette-1 scanned ionograms. The results are extracted mapped coordinates (i.e: digital ionograms), along with their read metadata (i.e: the ground station that telemetry was received, date and time of topside sounding, etc).

Metadata for each fully processed and read image are found on the FTP server, here:

- /users/OpenData_DonneesOuvertes/pub/Alouette-ISIS/Alouette-1/result/result_master.csv

Information about images that were not fully processed or read, can be found in the intermediate results tables (‘result_postprocess_raw.csv’, ‘result_stage2_raw.csv’, ‘result_stage1_raw.csv’).

The extracted mapped coordinates, as described in section III, are found here:

- /users/OpenData_DonneesOuvertes/pub/Alouette-ISIS/Alouette-1/result/mapped_coords/

For details about the processing event, the telemetry stations, and other information about the scanned images, can be found here:

- /users/OpenData_DonneesOuvertes/pub/Alouette-ISIS/Alouette-1/log/

Alouette-1 ionograms have three different metadata types, as described in Naidoo (2023) and Lessard (2017), and which are depicted in Figures 6 and 7.

The digitization of Alouette-1 film rolls occurred in two batches. ‘BATCH I’ scanning started in 2017, while ‘BATCH II’ started in 2021.

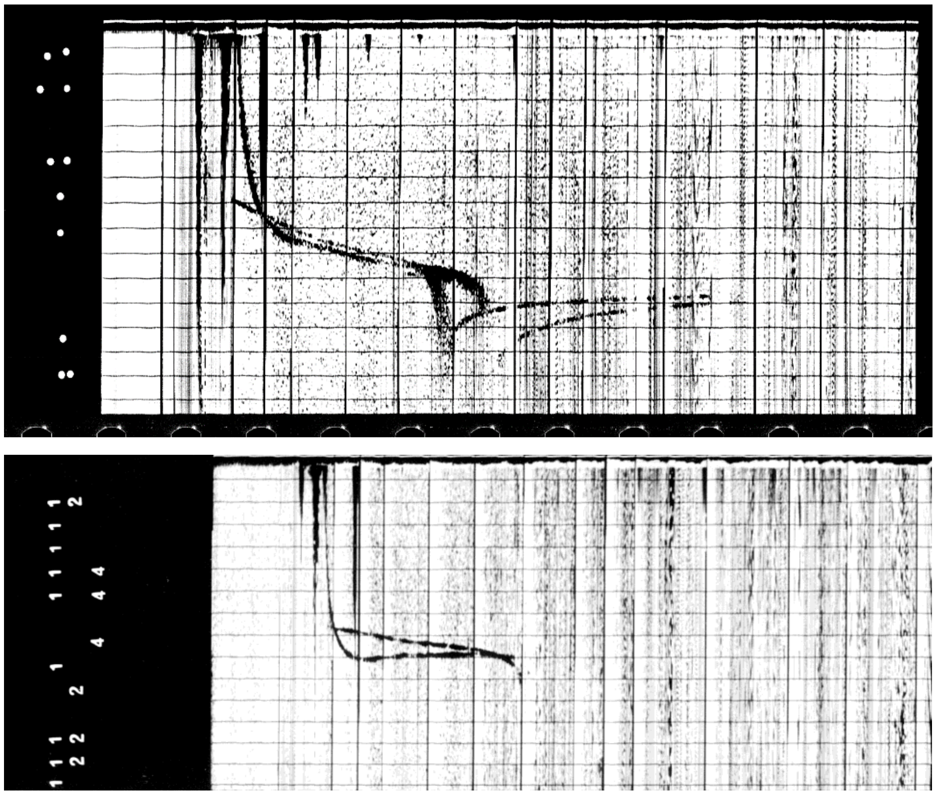


Figure 6: (top) ‘dot’ - Ionogram with dot metadata type. Early ionograms produced by DRTE encoded metadata into horizontal lanes, where each lane would represent a particular digit. The horizontal positions of dots within each horizontal lane would encode a number in binary format; (bottom) ‘num’ - Due to the ambiguity of the relative positions of each dot in each metadata lane, dots were replaced with numbers, but they were still arranged in the binary format. Therefore, one lane with a 1, 2, and 4 would be summed and represent 7. The new metadata format also included a lane to represent the year.

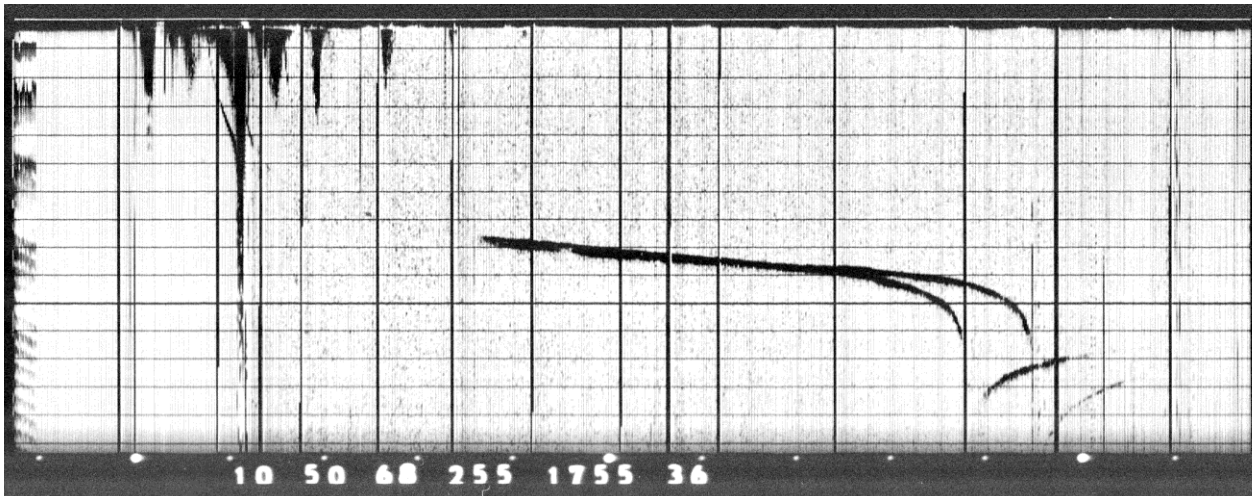


Figure 7: 'num2' - An easier to read numeric metadata format was introduced in 1965. A machine-learned text recognition model was used to read the metadata from ionograms with this format.

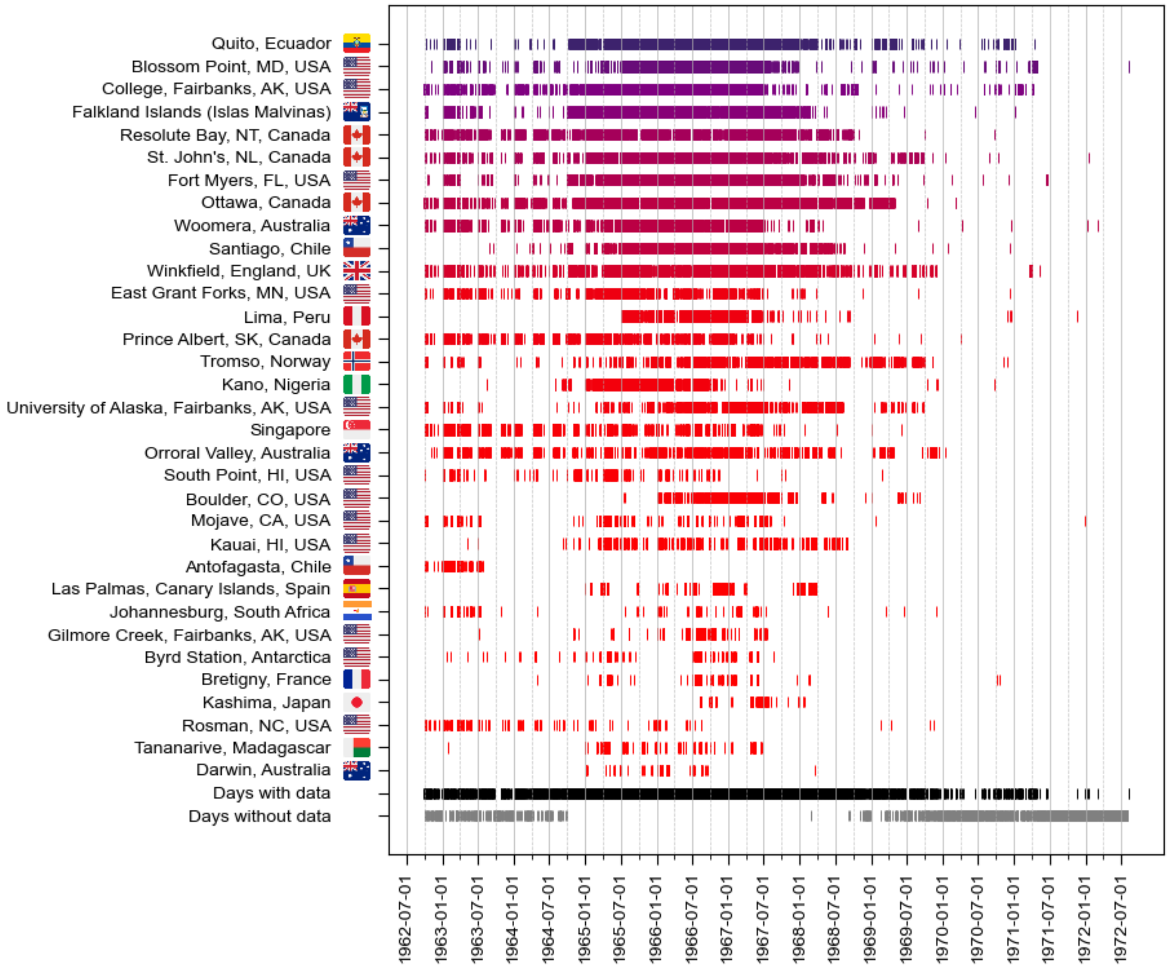


Figure 8: Current availability in days of Alouette-1 extracted data, by ground station.

The total availability of fully extracted Alouette-1 data across time, and by ground station, is depicted in Figure 8.

VI. Quality Analysis

Ultimately, **693,677 scanned images were fully processed and read** – representing a massive new dataset of Alouette-1 digitized topside ionograms.

However, out of 1,612,104 available Alouette-1 scanned ionogram images, **43.03%** were thought to be fully processed and read (i.e: the yield). Though as this section will discuss, there is some degree of error in data extraction and metadata reading within the set of 'fully processed and read' images.

As detailed in section VI, subdirectories of scanned images are processed through two stages (Stage 1, Stage 2), and then undergo a post-processing step to produce the 'result_master' dataset. After Stage 1 processing, 75.64% of images were fully processed, with 24.36% lost for various reasons (see Table 1). After Stage 2, the proportion of images that were considered fully processed reduced from 75.64% (after Stage 1) to 58.44%, for various reasons related to metadata misreading (see Table 2). After Stage 2 processing, 44.81% of images were classified as 'num', 9.06% were classified as 'num2' and 4.57% as 'dot' metadata type.

Table 1: Top-5 Functions That Caused Loss – After Stage 1 Processing

Function	Images
image_segmentation.segment_images_in_subdir.segment_images: iono size outlier	81,777
metadata_translation.determine_leftside_metadata_grid_mapping.extract_centroids_and_determine_type	38,122
ionogram_content_extraction.extract_select_parameters.extract_fmin_and_max_depth	22,546
image_segmentation.trim_raw_metadata.trimming_metadata	10,550
image_segmentation.segment_images_in_subdir.segment_images: metadata size outlier	2,731

Table 2: Top Recorded Reasons for Loss – After Stage 2 Processing

Detail	Images
metadata could not be read by OCR	246,360
OCR read metadata contains letters	23,984
metadata was interpreted to be num type	4,087
metadata was interpreted to be dot type	4,073

In the post-processing step, further losses could be attributed to various reasons. For the 'num' and 'dot' metadata types, it is assumed that each metadata lane must have a range of 0 to 10. Therefore, in cases where a metadata lane was read to be > 10, the result was overwritten as 'NaN' (not a number). Then as metadata digits were combined to form the various metadata elements (i.e: year, day of year, hour, etc), if these exceeded logical bounds (such as day of year > 366), then the metadata element was also rendered as 'NaN'.

As a result, the timestamp associated with each ionogram could be rendered 'NaT' (not a time) if just one time element was 'NaN'. Therefore, a 'time quality' level was ascertained for each processed and read image in 'result_master'. A time quality of 1 means that the time is read to the nearest second (i.e: all time elements were read); time quality 2 – to the nearest minute; time quality 3 – to the nearest hour; time quality 4 – to the nearest day. In 'result_master' while the overall yield is 43.03%, the yield with at least time quality level 4 is 44.50%.

Finally, if a station number is read but it does not correspond with a known ground station, then this will also result in a loss. Further details about the general quality analysis can be found here:

- **04_Quality Analysis - General**

Processing yield is also limited by scan quality issues, film quality issues, and even telemetry tape quality issues. For instance, the NASA Goddard data restoration project documented that when reading the telemetry tape, there were sometimes more frequency markers detected than expected, and the ionogram frame sync pulse (that indicates the start of the ionogram) was sometimes not detected at all. These telemetry tape data quality issues would have likely impacted the analog ionograms produced in the 1960s, and such issues are observed in the scanned images today.

Therefore, it is often observed that the quality of extracted data from an image can be related to its subdirectory, where there are entire film rolls (subdirectories) that can have poor quality. As Figure 9 depicts, there are almost 700 subdirectories that have a very poor yield (< 10%).

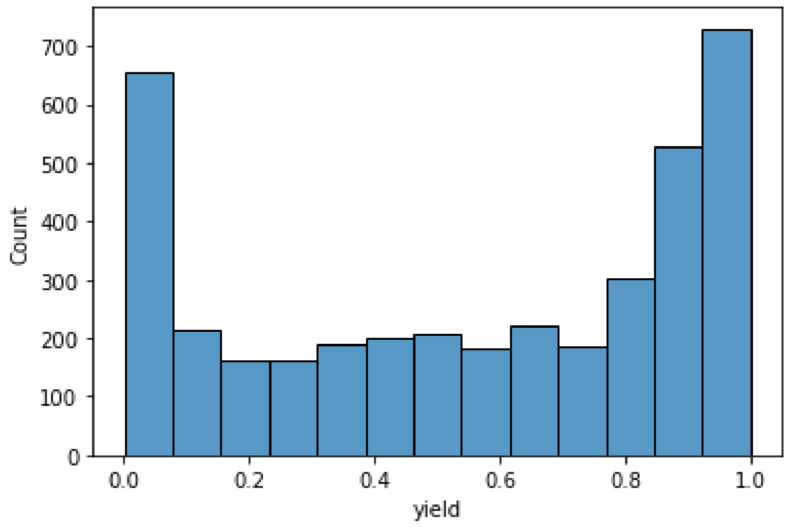


Figure 9: Histogram of yield in result_master dataset, by subdirectory.

5.1 Scan Error Detection

In the digitization of Alouette-1 35 mm film rolls, several issues arose that could be attributed specifically to the scanning of the film and the production of images. While it would require a great manual effort to check each subdirectory of images and make a qualitative assessment, two specific issues could be detected programmatically.

‘Out-of-phase’ images were those that spanned parts of two adjacent ionograms (see Figure 10). This type of scan error can be detected by checking the aspect ratio (height:width) of the image and detecting the number of digits in each image (in the case of ‘num2’ type images). A normal ‘num2’ image has an aspect ratio between 2 and 3, and has 15 digits on the bottom. Images with < 13 or > 17 digits are flagged as abnormal and potentially ‘out-of-phase.’

An abnormal aspect ratio, < 2 or > 3 could be indicative of an ‘out-of-phase’ image, or in many cases they

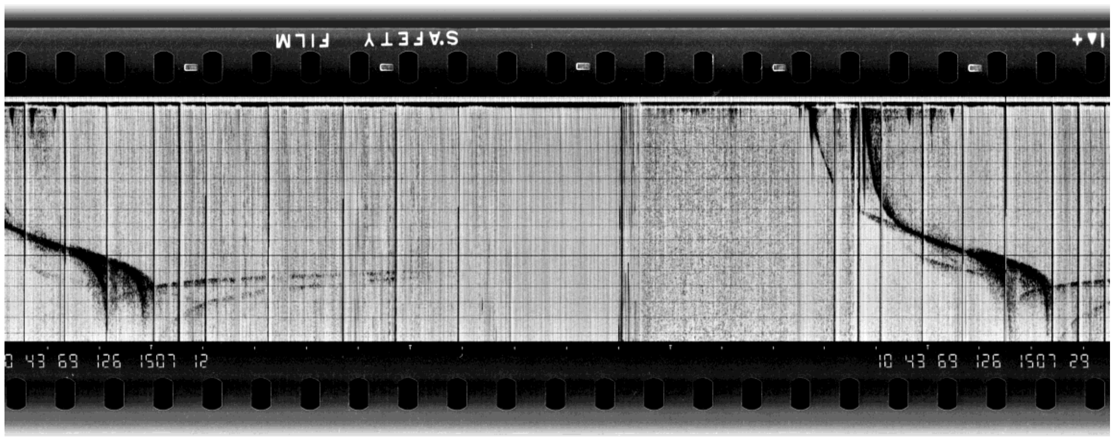


Figure 10: ‘Out of phase’ scanned image. The scan window should have been aligned to the start of the ionogram, in order to capture an entire ionogram, instead of parts of two adjacent ionograms.

are ‘irregularly cropped.’ It was found that some images were not cropped properly to display an entire ionogram (see Figure 11).

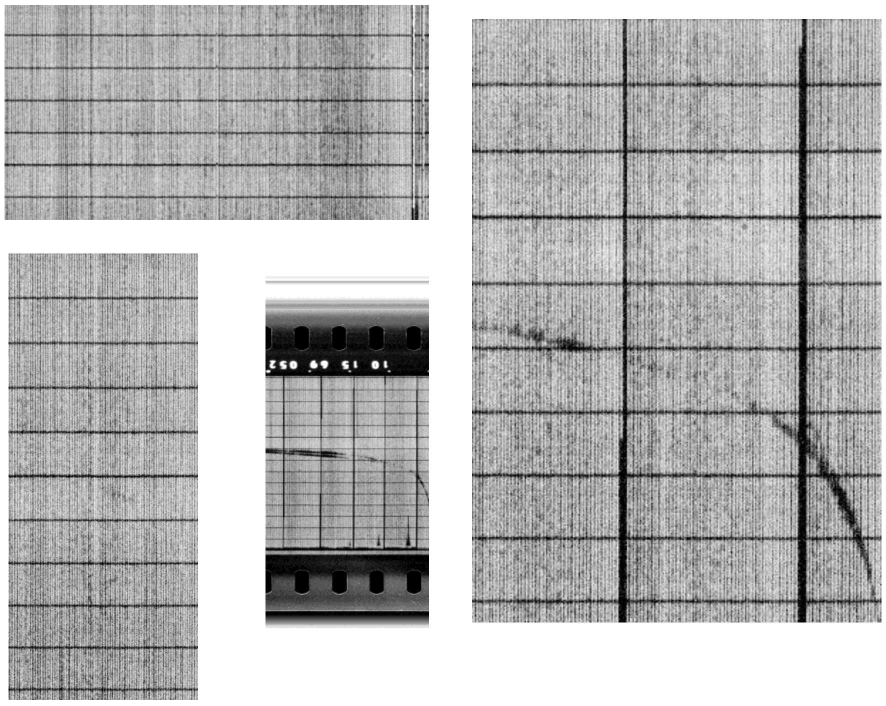


Figure 11: Examples of irregularly cropped images.

Further details about scan error detection can be found here:

- A_Scan Error Detection Analysis.ipynb
- Aa_Scan Error Detection (BATCH II).ipynb

- Ab_Efficient Scan Error Detection (BATCH II).ipynb
- Ac_Scan Error Detection Results Analysis.ipynb

By this method, for Alouette-1 images it was determined that about 4.7% of subdirectories may be out-of-phase and 0.1% were irregularly cropped.

5.2 Ionogram Mapping Accuracy

The data extraction algorithm attempts to read the metadata from a scanned ionogram image (see section 5.3) and extracts a set of datapoints that generally correspond to the ionogram trace (see Figure 2). The algorithm maps each datapoint from a (x, y) coordinate system of the image to a (frequency, depth) coordinate system, based on the reference lines on the ionogram (i.e: the grid system). The extraction process does pick up what are likely non-physical artifacts and noise that exist on the ionogram films, that were a result of scanning, or because of the data extraction itself (e.g: thick reference lines could be extracted).

Work was done to estimate the mapping accuracy of the data extraction algorithm, by drawing a random sample of 553 ionograms (399 images from BATCH I, 154 images from BATCH II) and manually reading the frequency and depth of the ionogram at the maximum-depth point. This is compared to the maximum-depth point from the corresponding set of extracted mapped coordinates (see Figure 12).

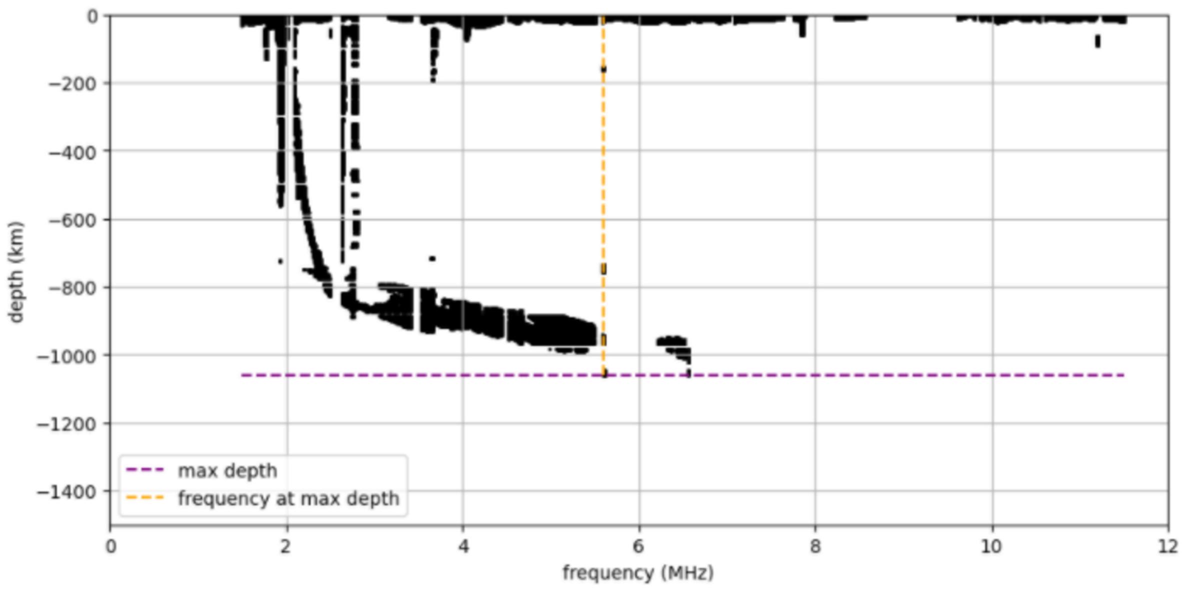


Figure 12: Maximum-depth point shown on a plot of extracted mapped coordinates.

It was determined that for BATCH I, the extracted maximum-depth point was the same (within ± 1 MHz, and ± 100 km) as what was manually read just over 50% of the time. However, BATCH II showed significantly worse results with mapping being the same as what was manually read only ~ 10% of the time. This may be because ionogram quality was generally worse for BATCH II, as determined by the scan error analysis (see section 5.1) and qualitative observations. From these results one may conclude that the performance of the ionogram mapping aspect of the data extraction algorithm is strongly dependant on the quality of the ionogram images.

Details about mapping accuracy determination can be found here:

- B_Determine Data Mapping Accuracy.ipynb

The usability of the extracted ionogram points will be limited by the type and rigour of scientific analysis required. However, taken in aggregate a large volume of extracted ionogram traces as these could still serve valuable purposes for scientific research and the Canadian public, as illustrated in Naidoo (2023).

Further work can be done to improve the extraction of the ionogram trace(s) while discarding line-based noise and other artifacts.

5.3 Metadata Reading Accuracy

As shown earlier, Alouette-1 ionograms have metadata encoded in one of three metadata types: 'dot', 'num', and 'num2'. A random sample of images for each metadata type was drawn. The metadata for each image was then manually read and checked against what was read by the data extraction algorithm (see Table 3).

Table 3: Metadata Reading Accuracy of Random Sample of Alouette-1 Ionograms, by Metadata Type

Metadata Type	**Images, Stage 2 processed** (BATCH I, BATCH II)	**Images, manually read from random sample**	**Images, entirely correctly read by algorithm from random sample**	**% of images correctly read by algorithm from random sample**
num	579,054 (465,009, 114,045)	99	94	**94.9%**
num2	126,073 (38,149, 87,924)	49	42	**85.7%**
dot	12,265 (7,784, 4,481)	25	0	**0.0%**

The metadata reading accuracy for each metadata type random sample is reported as the proportion of images *entirely* read correctly by the data extraction algorithm. In the num type random sample, some images were misread because they were low quality num2 images; the algorithm was unable to correctly determine that the image was of num2 type, and so it was misclassified as num type. In the num2 random sample, some errors were due to the text recognition model incorrectly reading the number 1 as 7, and vice-versa. Or it could misread 6 as 8, and vice-versa. Finally, dot type metadata reading by the data extraction algorithm is inadequate and should be improved in future work. This is understandable as it is also difficult to manually read dot metadata.

Looking more specifically at the metadata reading accuracy for each metadata element, generally there does not appear to be a systematic error in the misreading of a certain metadata element – in either the num or num2 type images. See Figures 13 and 14 on page 13 and 14 respectively.

As for the dot type images, the reading accuracy for each metadata element was zero for the random sample taken. The algorithm could not easily differentiate between the positions of, and our team had difficulty with manually reading the dot type images to fine tune the code.

Further details regarding how the metadata reading accuracy was determined, can be found here:

- C_Overall Metadata Reading Accuracy for Alouette-1.ipynb

- [Ca_Metadata Reading Accuracy (BATCH I).ipynb]
- [Cb_Metadata Reading Accuracy (BATCH II).ipynb]

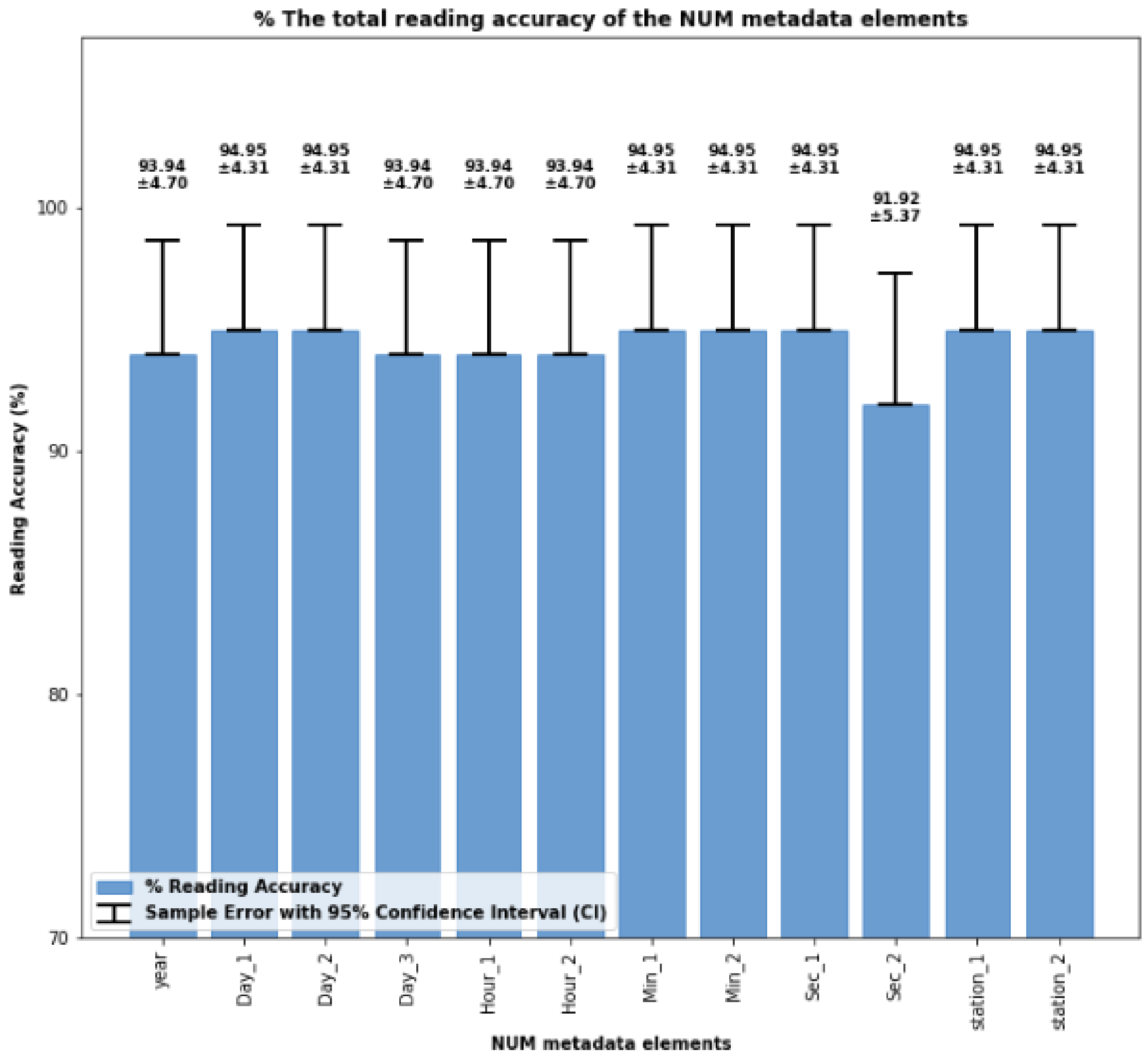


Figure 13: Metadata reading accuracy from random sample of num type ionograms, by metadata element.

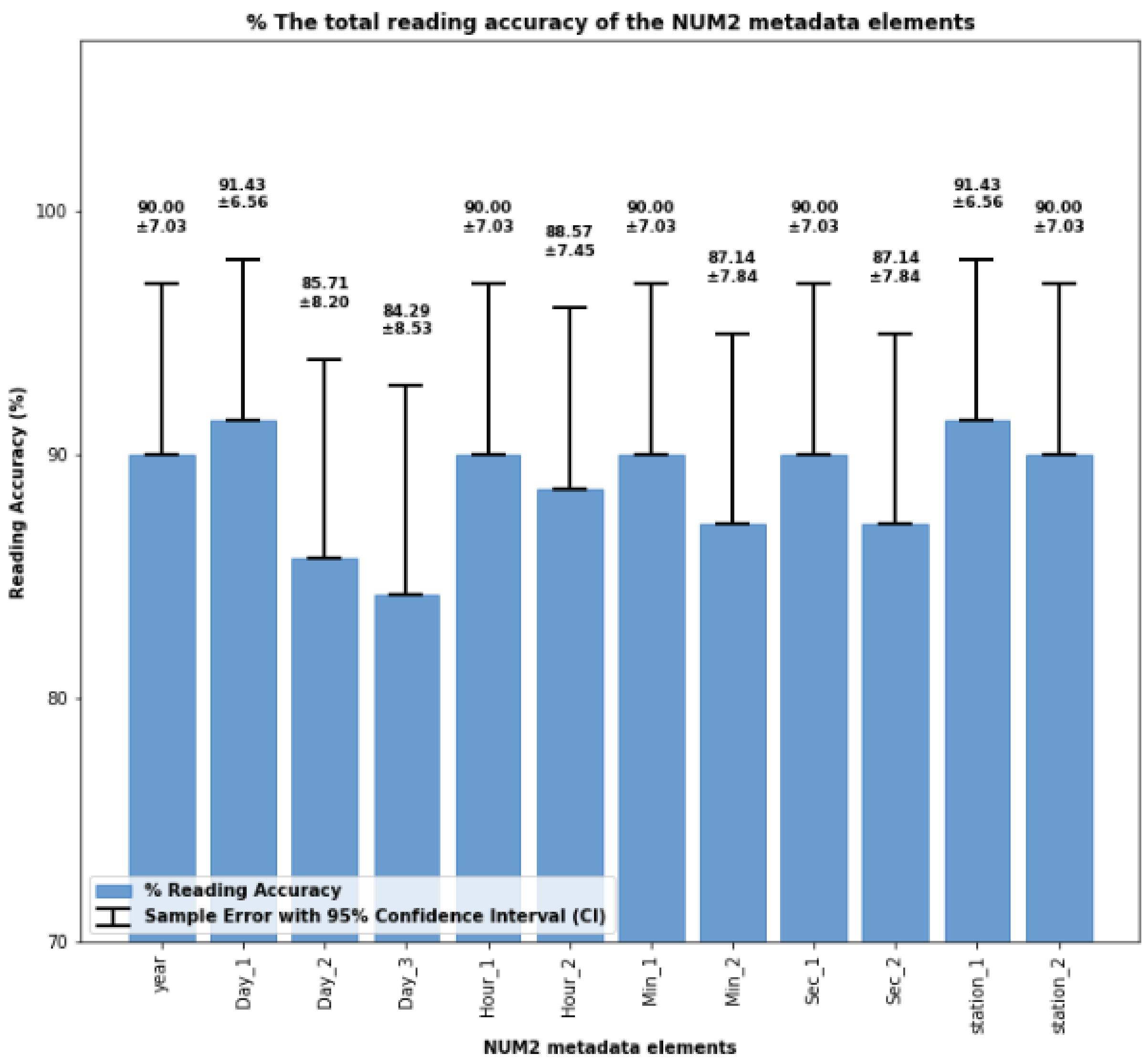


Figure 14: Metadata reading accuracy from random sample of num 2 type ionograms, by metadata element.

5.4 Orbit Check

In a further effort to cross-validate the read metadata, it is possible to compare this to the historical position of Alouette-1. For a given ionogram representing telemetry received at a particular ground station at a particular time, this can be checked against the estimated historical position of the satellite and whether it was within range of the receiving ground station (different configurations existed for the minimum altitude above the horizon from the ground station of 15 degrees to 0 degrees, with altitudes above this deemed ‘viewable’).

Satellites orbiting the Earth are tracked using various methods, and their positions are often represented using a standard format known as the Two-Line-Element (TLE). A TLE is a data format used to convey sets of orbital elements of an Earth-orbiting object. These elements are derived from observational data and are used to predict the object's future orbital path. A TLE consists of two lines of text, each containing specific details about the satellite's orbit.

An example TLE looks like this:

```
1   424U 62049A   23231.84816197  .00000113  00000-0  11921-3 0  9998
2   424  80.4670 197.6975 0023664  22.8904 337.3291 13.69292735 40017
```

Line 1 contains the following information:

- Satellite NORAD Catalog Number
- International Designator (Launch Year, Launch Number, Piece of Launch)
- Epoch Year and Julian Day Fraction
- First Time Derivative of the Mean Motion
- Second Time Derivative of Mean Motion (often zero)
- BSTAR drag term
- Ephemeris type (usually zero)
- Element set number (incremented when multiple TLEs are produced for the same object on the same day)
- Checksum (for error checking)

Line 2:

- Satellite NORAD Catalog Number (repeated)
- Inclination (angle between the satellite's orbital plane and the Earth's equatorial plane)
- Right Ascension of Ascending Node (RAAN)
- Eccentricity (shape of the orbit)
- Argument of Perigee (angle between the ascending node and the satellite's closest approach to Earth)
- Mean Anomaly (position of the satellite in its orbit at the epoch time)
- Mean Motion (number of orbits completed in one day)
- Revolution number at epoch (how many times the satellite has orbited the Earth since launch)
- Checksum (for error checking)

The TLE provides the necessary parameters to determine the position and velocity of a satellite at a given epoch time. By using these parameters and applying the laws of celestial mechanics (specifically, the SGP4/SDP4 orbital models), one can predict the satellite's position and velocity at any other time.

Overview of the step-by-step process used:

1. Match the nearest TLE record to the read epoch of an ionogram’s metadata.
2. Determine Alouette-1’s position using the TLE parameters at the read epoch.
3. Determine the positional characteristics of the orbit including the altitude and orbit pass events in relation to the read receiving ground station from the metadata.
4. Flag metadata records that were not above the horizon and/or had nearby orbital passes to the read receiving ground station.

The orbital cross-check was calculated for all fully read and processed Alouette-1 ionograms. The results were derived for each ionogram metadata type (dot, num, num2). The number of ‘num’ records that were historically viewable by its receiving ground station was 57.4%, and for ‘num2’ it is 79.2%.

Several factors could explain the resulting estimated accuracies. Firstly, as it is a cross-check, flagged records could potentially signal an incorrectly read metadata element (either the epoch (the read ionogram datetime) or receiving ground station). Alternatively, as this means of calculation is an estimation, the estimation itself could be incorrect. Several factors could contribute to an incorrect estimation, including potential errors in the TLE’s themselves (less likely), level of accuracy of early TLE records (moderately likely), decreasing accuracy of the matched read epoch to the TLE record (most likely).

Further details regarding how the orbit check was conducted and processed, can be found here:

- **E_Satellite Track Processing.ipynb**

5.5 Data Dictionary

The data dictionary is a centralized reference for the understanding of and managing of data elements within a given dataset. It provides clear descriptions, data types and relationships between the data elements. The data dictionary's framework is heavily influenced by the PDS4 standards. This standard is prevalent in planetary science and space research fields. The PDS4 standard provides robust flexibility for several data domains, and interoperability. This promotes a standardized common framework for data communication.

Limitations & Future Work

- **Current Constraints**:
  - Data Format Support: The processing as outlined for creating a data dictionary is currently confined to handling CSV and Excel file formats (or other structured table formats). This is due to other data formats each being handled with a custom process.
  - User Input Dependence: The current process relies significantly on manual user input, for specification of data types, units, and descriptions.
- **Proposed Enhancements**:
  - User Interface: Enhancing the user interface with more interactive and intuitive elements could improve interpretability.
  - Error Handling: Introducing advanced error handling and validation checks would elevate the notebook's robustness and reliability to adhere to framework requirements.

Workflow

- **Steps**:
  - 1: Prepare a python virtual environment (>3.5 version) with the required libraries as specified in the notebook.
  - 2: Note the file path to the structured table data to be processed in creating a data dictionary.
  - 3: Using the Jupyter Notebook, run the cells in sequential order and follow the user prompts. The data dictionary will be saved to the current directory.
  - 4: If multiple data dictionaries are associated with a given data bundle, copy the dictionaries' paths into the final cell of the Jupyter Notebook. This creates a sheet within a structured table workbook (Excel) for each individual data dictionary. Create an additional sheet as an overview to reference the individual data dictionaries.

Data Dictionary Description

- **Column Descriptions**:
  - `Column_Name`: This defines the extracted column name for the processed structed data table. This provides reference to the individual column to access the specific data element.

   - `Description`: This defines the user provided description for the column of the structured data table. This description should be easily interpretable and provide sufficient information or reference to the characteristics of the data in the column.
   - `Data_Type`: This defines the characteristics of the column of the structured data. Further sub-types are specified when required to further define the characteristics of the data. The data types used are based on the PDS4 types available [here].
   - `Stats`: This defines additional statistics that are calculated for certain data types. The statistics calculated are in order of min, max, and size of the given column of data.
   - `Units`: This defines potential units of measurement for the individual column of the structured data table. The units available are based on the PDS4 units available [here].

Implementation Details

- **Data Name Definition**:
  - Function: The notebook incorporates a `get_column_name` function, enabling the extraction of the name of each dataset column.
  - Options: It extracts the column name for the provided dataset column.
- **Data Description Definition**:
  - Function: The notebook incorporates a `get_user_description` function, enabling users to describe or reference the unique features, characteristics, and information of each dataset column.
  - Options: It offers the user a prompt that allows for text base input. The user can provide any information that is relevant to the description of the data properties.
- **Data Unit Definition**:
  - Function: The notebook incorporates a `get_unit_selection` function, enabling users to assign specific data measurement units to each dataset column.
  - Options: It offers a range of data units available for the user to select. In the event that no unit is specified or available, the unit type defaults to `Unit_of_None`.
- **Data Type Definition**:
  - Function: The notebook incorporates a `get_data_type` function, enabling users to assign specific data types to each dataset column.
  - Options: It offers a range of data types such as Date_Time, File_Name, Numeric, Identifier, String, and Misc. Each primary category is further divided into more specific types for detailed classification. Each primary category and sub category have unique defaults in the event that the specified or available data type is invalid.
- **Data Statistics Calculation**:
  - Function: Utilizing the `get_data_stats` function, the notebook calculates essential statistics for each column.
  - Metrics: This includes determining minimum, maximum, and count values, tailored to suit various data types including numeric, identifier, and string.
- **File Analysis Process**:

    - Function: Central to the notebook is the `analyze_file` function. This function is responsible for reading the input file, extracting and processing column information, and then sequentially prompting the user to provide descriptions, select data types, calculate statistics, and choose units.
    - Result Compilation: It compiles all gathered information into a singular Pandas DataFrame. This DataFrame can then be saved as a unique data dictionary.
- **Execution and Output**:
    - User Interaction: The notebook is structured to allow easy listing of files in the current directory and facilitates the analysis of a chosen file.
    - Final Output: Upon completion of the analysis, the results are conveniently saved into a CSV file, enhancing practicality for application in various settings.

Further details regarding how the orbit check was conducted and processed, can be found here:

- **G_Alouette Data Dictionary.ipynb**

VII. Re-Processing

For many purposes the already processed data may be sufficient. However, one may wish to re-process the images, and even make improvements upon the current data extraction method.

5.6 Image Inventory

The image inventory is required for the processing method presented. It is already provided, here:

- /users/OpenData_DonneesOuvertes/pub/Alouette-ISIS/Alouette-1/log/image_inventory.csv

However, if somehow new scanned images were to become available, or for any other reason, the image_inventory.csv can be regenerated by running:

- **02_Image Inventory.ipynb**

5.7 Stage 1 Processing

Subdirectories of scanned images are processed through two stages: ‘Stage 1’ and ‘Stage 2’. Stage 1 is the original, or legacy, data extraction algorithm that was developed from 2017 – 2020. A wrapper was developed around the original algorithm to manage and log performance of this processing step, along with other improvements.

Stage 1 processing can occur by running multiple instances of Alouette_processor.py. You can run as many instances as your system will allow. Multiple systems can even be used to run instances, so long as the root directory for the project is in a shared location. This is in order to reduce the processing time.

Details about setting up the root directory are in ‘00_Download a Subdirectory from FTP.ipynb’. Ensure that the root directory has the following folders:

- 04_processed
- 04a_unprocessed
- 05_result
- 06_log

Furthermore, set up an instance directory. If only running one instance, the instance directory can be the root directory. If running multiple instances, set up a separate instance directory for each instance (e.g: 'Stage1_instance1/', 'Stage1_instance2/', etc). In either case ensure that each instance directory has the following folders:

- 03_processing
- 05a_result_local

To start Stage 1 processing, run:

```
Alouette_processor.py [the file path to the scanned images, organized
into Directories and Subdirectories] [the file path to root directory
for your project] [The file path to the instance directory] [username
(for logging purposes)] [instance number]
```

For example:

```
python Alouette_processor.py “C:/Projects/Alouette-1/data/”
“C:/Projects/Alouette-1/root/” “C:/Projects/Alouette-
1/Stage1_instance_12/” “UserX” 12
```

A process_log.csv is produced and saved in the 06_log folder. This is helpful to keep track of processing, and what is accessed by multiple instances to ensure a subdirectory is not processed more than once. It is also useful to analyze the processing performance. If one is processing the entire Alouette-1 image set, be prepared to run multiple instances, and perhaps multiple systems, to bring down the projected processing time required. Processing with 12 instances of Alouette_processor.py on a high-performance machine with a 24-core CPU and 128 GB RAM still took two to three days to process.

After Stage 1 processing is complete, open:

- **03_Post-Processing.ipynb**

In this Jupyter notebook, only run the first section:

> ***Generate processed_inventory, concatenate Stage 1 results - RUN BEFORE starting Stage 2 processing (OCR processing stage)***

This will concatenate the Stage 1 results for each Subdirectory, and save it as ~/05_result/result_stage1_raw.csv

5.8 Stage 2 Processing

The Stage 2 processing step is relatively more straightforward. This stage was developed in 2023 to apply a machine-learned text recognition model (keras_ocr) to detect 'num2' type images, and then read that form of metadata.

The keras_ocr model used has a convolutional neural network (CNN) to first locate text on a given image. Then, the text is read with a text recognition classifier. The model's use of the CNN can be a computationally intensive process. Therefore, a GPU version of the code was developed for this Stage, in addition to a standard CPU version. Leveraging a system's GPU enabled this stage to progress significantly faster.

To start Stage 2 processing, run:

```
Alouette_processor2.py [the file path to root directory for your
project] [username (for logging purposes)] [instance number] [batch
size]
```

For example:

```
python Alouette_processor2.py “C:/Projects/Alouette-1/root/” “UserX” 1
8
```

And, to run the GPU version:

```
Alouette_processor2_gpu.py [the file path to root directory for your
project] [username (for logging purposes)] [instance number] [batch
size]
```

Because it is a computationally demanding process, it was found that only 1 CPU instance and 1 GPU instance could be run before system resources were fully utilized – though your system specifications may allow for more instances. The batch size can also be varied based on your system specifications. However, if the batch size is set too large, the program will crash.

5.9 Post-Processing

After Stage 2 processing is complete, open:

- **03_Post-Processing.ipynb**

Now run the remainder of the script, starting from:

Concatenate 'OCR pass' results - START HERE after Stage 2 processing (OCR processing stage)

This step will process the datetime, retrieve the ground station details, among other things, to save the ~/05_result/result_master.csv file.

6.1 Quality Analysis

General quality analysis and statistics can be produced after post-processing, to determine the yield, and other things.

Run:

- **04_Quality Analysis - General**

VIII. Future Work

Many things could be done to improve the ionogram data extraction yield, which is currently **43.03%**. Nevertheless, there will always be a limit to the yield because of scan quality, film quality, or quality issues with the original telemetry tape.

At least two improvements to the data extraction method could significantly improve the yield. First, it appears that the Stage 1 processing step is not able to reliably read the 'dot' metadata type, if at all. In the least, the mapping of the relative position of dots to binary metadata in each metadata lane should be checked and recalibrated. Or the entire process ought to be rebuilt to be able to detect and read dot type scanned ionogram images. While dot type images may represent a relatively small proportion of the total set of Alouette-1 scanned images, they occur within the first year of operation of Alouette-1. The early years (1962-65) of Alouette-1 have the greatest scientific interest, as they are the earliest and only topside soundings of the ionosphere made by humanity, until the launch of Alouette-2 on November 29, 1965.

The default keras_ocr text recognition model that was used could be further trained to better read 'num2' metadata numbers, and to only read numbers and not letters. For instance, there were at least 23,984 num2 images that seemed to adhere to the metadata format (i.e: had 15 characters beginning with '10'), but then had certain numbers misread as letters. There are also several instances where an image that is actually of num2 type was not read, perhaps due to scan quality issues. Further training of the text recognition model could improve the reading accuracy.

As described, the data extraction method occurs in two stages – to first run the old data extraction algorithm, and then apply the new text recognition stage to read num2 metadata. The algorithm should be refactored such that the metadata type can be detected initially, and then read according to its type. This refactoring and rebuilding of the code base could resolve certain long-standing issues and may improve yield. Furthermore, it can enable the algorithm to be more adaptable to be able to read Alouette-2, ISIS-1 and ISIS-2 ionograms.

Yet with some modifications, ISIS-1 and ISIS-2 ionograms seem to have been read successfully.

Prepared by: Ravendra Naidoo, Roksana Sheikholmolouki, Ashley Ferreira, Marianne Fortier, Jackson Cooper, Benjamin Cannings, Émiline Filion

Last Revision: February 5th, 2024