NASA + Advanced Air Mobility (AAM)
National Campaign (NC) Tech Talk: Integrated Data Product

Timothy Bagnall and Ledger West | ATI Data Services Team
NASA Aeronautics Research Mission Directorate | Sep 29 2023

NASA AAM NC Technical Talks

- Purpose of these Tech Talks is to engage with the community on NASA technologies
- Ground rules:
  - Answers to questions you have may be in upcoming slides
  - It is okay to ask an important question on a slide, but please wait until the end, if possible
  - Mute your mic unless you need to talk
  - NASA will keep a questions parking lot to keep the Tech Talk on point and on time
- NASA is recording this Tech Talk for future viewing

Agenda

- National Campaign Overview (Nicole)
- Integrated Data Product Overview (Tim)
- Partner Tailoring (Tim)
- Access for Analysts (Tim)
- Technology Focus (Ledger)
  - Detailed description
  - Architecture
  - Technical tools / Stack
  - Data flow
  - Design pattern / Under the hood
  - Applications
  - Try it at home (three simple postflight data files)

National Campaign Overview

NASA's AAM vision:

- Safe, sustainable, accessible, and affordable aviation for transformational local and intraregional missions.
- Transportation of passengers and cargo as well as aerial work missions, such as infrastructure inspection or search and rescue operations.
- Local operations of about 50-mile radius in rural or urban areas, and intraregional operations of up to a few hundred miles that occur between urban areas, between rural areas, or between rural and urban areas.

The AAM National Campaign (NC) is designed to:

- Promote public confidence in AAM safety.
- Give prospective vehicle manufacturers and operators, as well as prospective airspace service providers, insights into the evolving regulatory and operational environment.
- Facilitate community-wide learning while capturing the public's imagination.

NC has already participated in over a dozen state-of-the-art flight tests!

National Campaign Overview

NASA NC Partners have included:

wisk

RELIABLE ROBOTICS

Joby

SIKORSKY

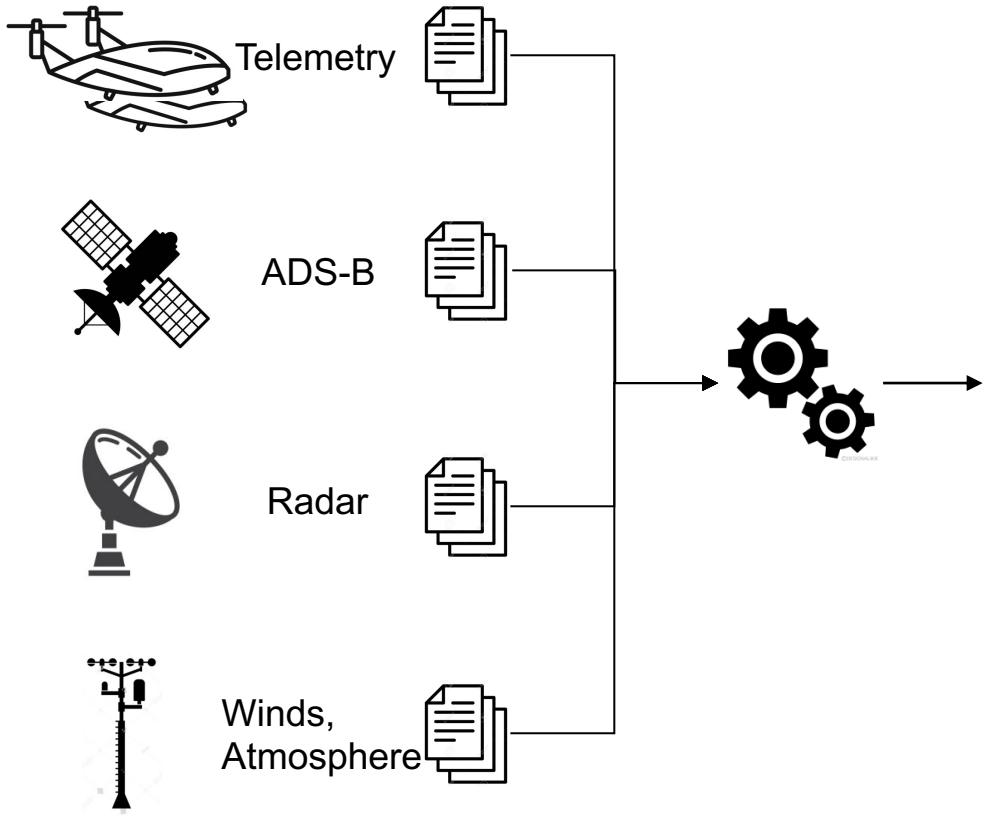
IDP Overview

- The AAM NC *Integrated Data Product* (IDP) is a suite of software and infrastructure that combines data produced by disparate flight test instruments, both ground-based and airborne, into a single, analysis-ready product.
- IDP processing:
  - Creates consistent time synchronization across all data sources
  - Detects and addresses missing or invalid data
  - Enforces naming conventions and unit consistency
  - Has a flexible workflow that is configuration driven
  - Is tailorable for each flight test activity
  - Is additive – new capabilities developed for one activity are available to future tests
- IDP infrastructure enforces data protection for NASA and partners
- IDP furthers AAM/AMP research objectives by providing a single, easily usable source of vehicle(s), airspace, and environmental data for any desired analyses.

“You cannot have information without data.” — Daniel Keys Moran

IDP Overview cont.

An IDP results from processing of all flight test data sources into single, tabular, time series product.



synch_utc	Lat	Long	Alt	Wind	Gust	...	...	...
2021-04-11 16:06:22+00:00	37.368	37.662	2593.992	3.700	9.500	...	...	...
2021-04-11 16:14:01+00:00	37.975	38.026	2590.224	4.692	10.137	...	...	...
2021-04-11 16:17:56+00:00	37.342	37.474	2588.387	5.183	10.418	...	...	...
2021-04-11 16:20:10+00:00	37.212	37.998	2594.823	5.652	10.628	...	...	...
2021-04-11 16:22:23+00:00	37.240	37.142	2593.232	5.758	11.257	...	...	...
2021-04-11 16:23:21+00:00	37.934	37.417	2592.317	5.947	11.527	...	...	...
2021-04-11 16:27:18+00:00	37.647	37.085	2595.007	6.874	11.678	...	...	...
2021-04-11 16:28:38+00:00	38.019	37.993	2590.295	7.216	12.189	...	...	...
2021-04-11 16:29:07+00:00	37.912	37.912	2594.805	7.676	12.832	...	...	...

IDP Overview cont.

- Below is an abbreviated, sample IDP having only nine fields
- IDPs typically having well over 100 fields, based on the goals and constraints of a flight test event and the host test range.
- For consistency, IDP fields have a standard naming conventions and units of measure
  - E.g., Latitude is always “Lat” and in decimal degrees; “altitude_msl” is feet above mean sea level
- IDPs often include multiple sources of the same datum (e.g., multiple sources for 3D position as illustrated by Source1 and Source2 below)

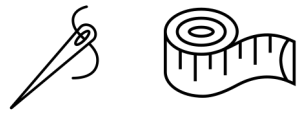
synch_utc	Lat [Source1]	Lat [Source2]	Lon [Source1]	Lon [Source2]	altitude_msl [Source1]	altitude_msl [Source2]	Hwindspeed [Source3]	Hwindspeed [Source4]
2021-04-11 16:06:22	37.368	37.662	77.947	77.986	2593.992	2593.981	7.392	7.380
2021-04-11 16:06:23	37.975	38.026	78.035	78.001	2590.224	2590.315	7.392	7.352
2021-04-11 16:06:24	37.342	37.474	77.965	77.969	2588.387	2588.347	8.734	8.934
2021-04-11 16:06:25	37.212	37.998	77.997	77.959	2594.823	2594.885	6.720	6.094
2021-04-11 16:06:26	37.240	37.142	78.007	78.042	2593.232	2593.234	6.048	6.435

IDP Overview cont.

- All NC partner and NASA flight test events are different!
- IDP content and composition varies by flight test event by adapting to research goals, installed instrumentation, and other limitations or constraints.
- IDPs are tailored in close coordination with NC partners
- The partnership defines the data to be collected during a flight test event, including such details as:
  - Data sources
  - Data source file names
  - Data fields
  - Data reporting frequencies
  - Units of measure

IDP Overview cont.

- NC Partners define fields targeted for collection during a flight test
- Note how source file names and source field names may differ



Standard IDP data fields

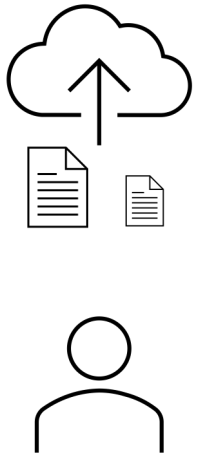
NC Partner provides details here

		NC Partner Input			
IDP Field Name	IDP Field Description	Expected Rate (Hz)	Filename	Field Name	Field Units
Position Velocity Time (PVT) Data					
lat	latitude	40	File A	latitude	DMS
lon	longitude	40	File A	longitude	DMS
altitude_msl	height mean sea level	40	File A	altitude_geoid	meters
altitude_hae	height above WGS-84 ellipsoid	40	File A	altitude_ellipsoid	meters
groundspeed	ground speed, knots	40	File A	speed	meters per second
track	ground track	40	File A	heading	degrees true north
Inertial Data					
pitch	aircraft pitch	60	File B	lateral	degrees. Up positive
pitch_rate	aircraft pitch rate	60	File B	lateral_rate	meters per second
pitch_rate_accel	aircraft pitch rate acceleration	60	File B	lateral_acceleration	g
roll	aircraft roll	60	File B	longitudinal	degrees. Right turn positive
roll_rate	aircraft roll rate	60	File B	longitudinal_rate	meters per second
roll_rate_accel	aircraft roll rate acceleration	60	File B	longitudinal_acceleration	g

IDP Access for Analysts

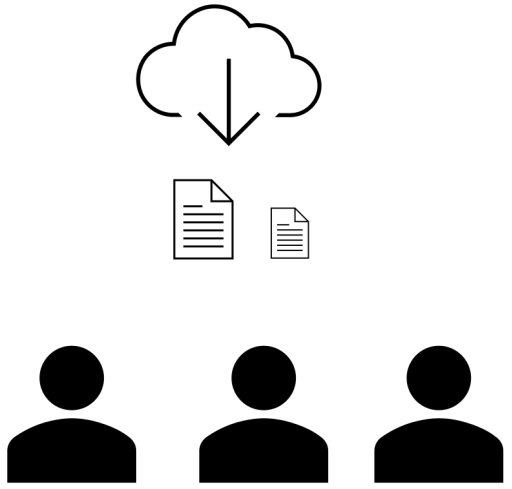
- Once an IDP has been produced, NASA uploads the files – in csv and parquet formats – to a secure and carefully controlled and administered Box folder
- Box is a cloud-based file sharing system
- Box is FIPS 140-2 certified, and every file is encrypted using AES 256-bit encryption at rest and in transit
- NASA's AAM NC data control team administers Box, vetting and qualifying all users

Partner A Box Folder



NASA Uploads IDPs to Box

Partner A Box Folder

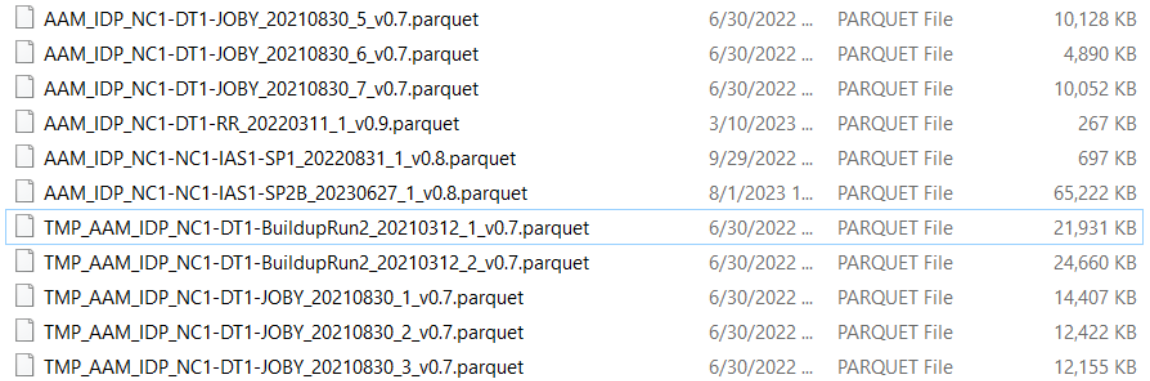


Qualified Users Download IDPs

IDP Detailed Description

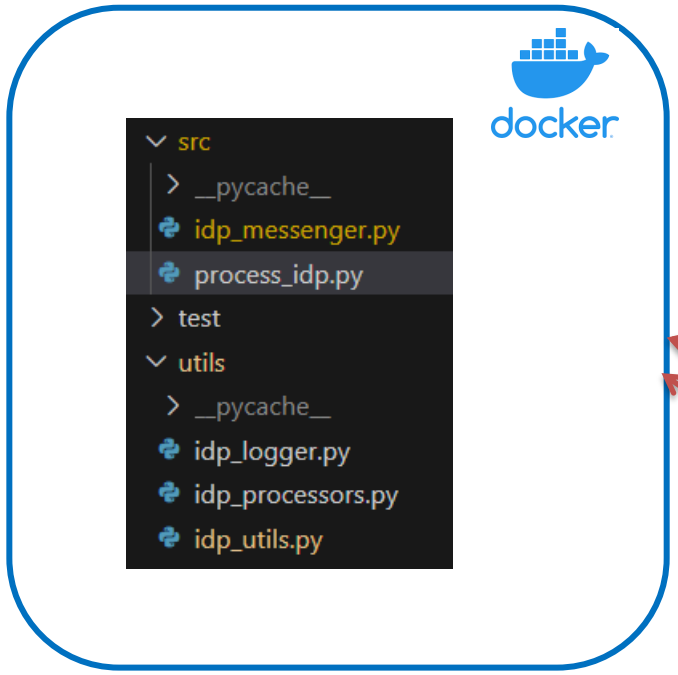
- ✓ IDP is a time-series dataset
- ✓ Term “IDP” also refers to ETL Pipeline that generates IDP
- ✓ Each IDP is event-based
  - ✓ Different sets of data
  - ✓ Custom transformations
  - ✓ Largely defined and governed by an Event Config file
  - ✓ Each event can yield multiple IDPs (across days and sorties)
- ✓ Maintains a concept of state
  - ✓ No assumptions about the order of receipt of input datasets
  - ✓ But within some events, some operations depend on certain datasets being present
  - ✓ A) process data as best it can as it arrives, and
  - ✓ B) make available “best effort” IDP versions without having to wait for all data

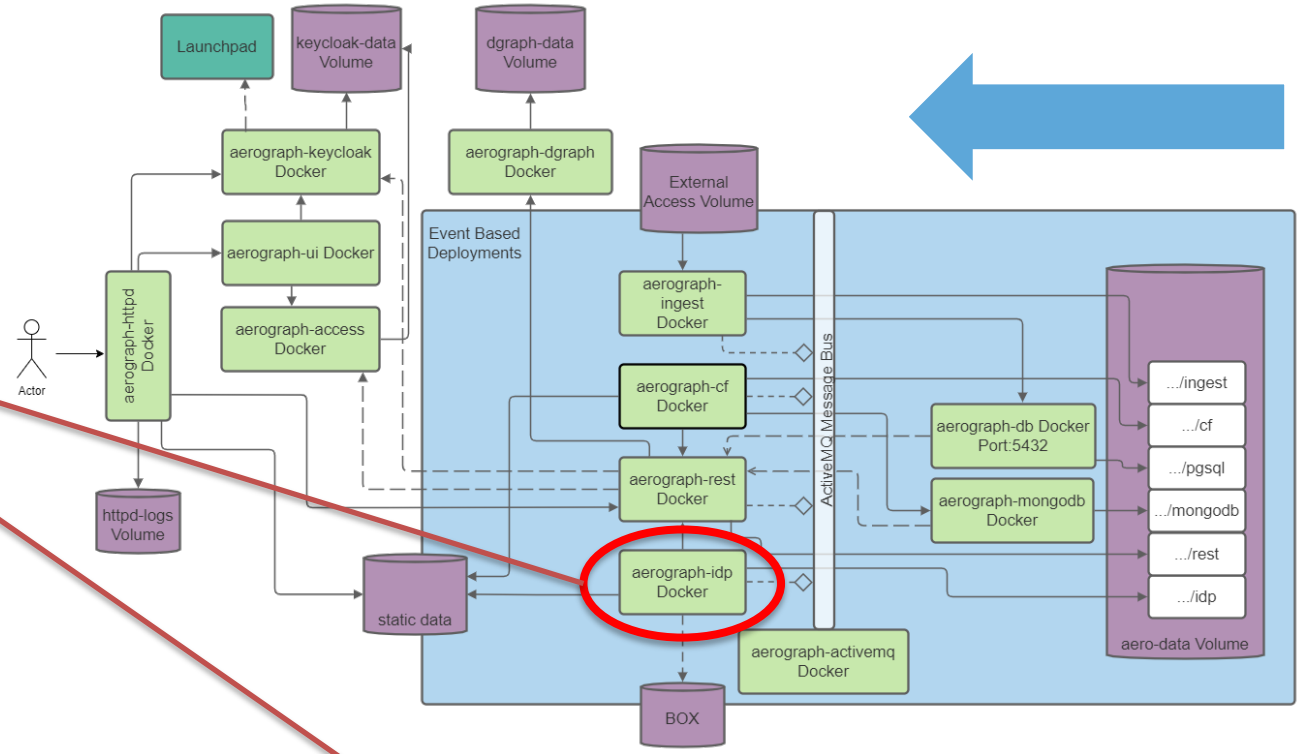
- ✓ IDP easily extends to new Events. Analyst will:
  - ✓ Create an Event Config file
  - ✓ Define computational functions (CF) for any new data sources
  - ✓ Add columns to a master Columns file
  - ✓ Add output column names and units of measure to an Output file
  - ✓ Use the naming convention for incoming datasets

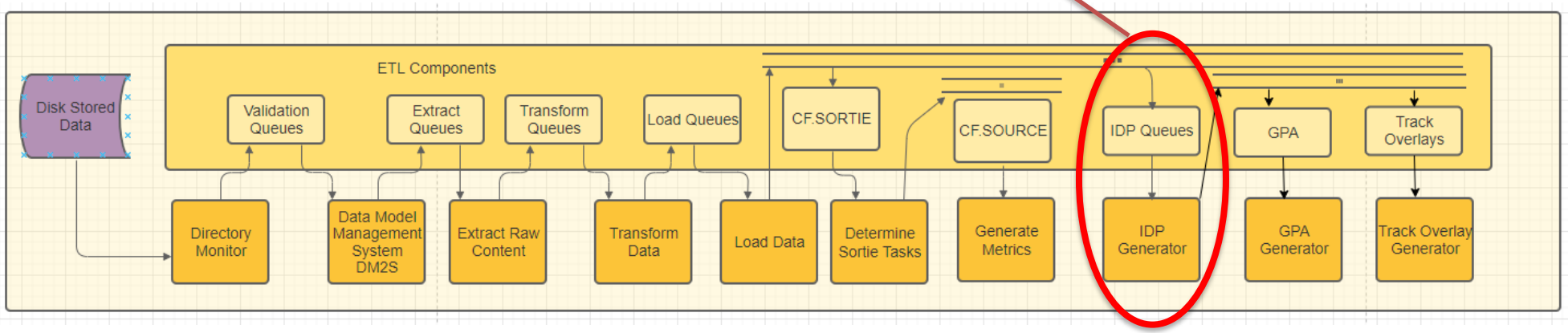


AAM_IDP_NC1-DT1-JOBY_20210830_5_v0.7.parquet	6/30/2022 ...	PARQUET File	10,128 KB
AAM_IDP_NC1-DT1-JOBY_20210830_6_v0.7.parquet	6/30/2022 ...	PARQUET File	4,890 KB
AAM_IDP_NC1-DT1-JOBY_20210830_7_v0.7.parquet	6/30/2022 ...	PARQUET File	10,052 KB
AAM_IDP_NC1-DT1-RR_20220311_1_v0.9.parquet	3/10/2023 ...	PARQUET File	267 KB
AAM_IDP_NC1-NC1-IAS1-SP1_20220831_1_v0.8.parquet	9/29/2022 ...	PARQUET File	697 KB
AAM_IDP_NC1-NC1-IAS1-SP2B_20230627_1_v0.8.parquet	8/1/2023 1...	PARQUET File	65,222 KB
TMP_AAM_IDP_NC1-DT1-BuildupRun2_20210312_1_v0.7.parquet	6/30/2022 ...	PARQUET File	21,931 KB
TMP_AAM_IDP_NC1-DT1-BuildupRun2_20210312_2_v0.7.parquet	6/30/2022 ...	PARQUET File	24,660 KB
TMP_AAM_IDP_NC1-DT1-JOBY_20210830_1_v0.7.parquet	6/30/2022 ...	PARQUET File	14,407 KB
TMP_AAM_IDP_NC1-DT1-JOBY_20210830_2_v0.7.parquet	6/30/2022 ...	PARQUET File	12,422 KB
TMP_AAM_IDP_NC1-DT1-JOBY_20210830_3_v0.7.parquet	6/30/2022 ...	PARQUET File	12,155 KB

IDP Architecture







IDP Technical Tools / Stack

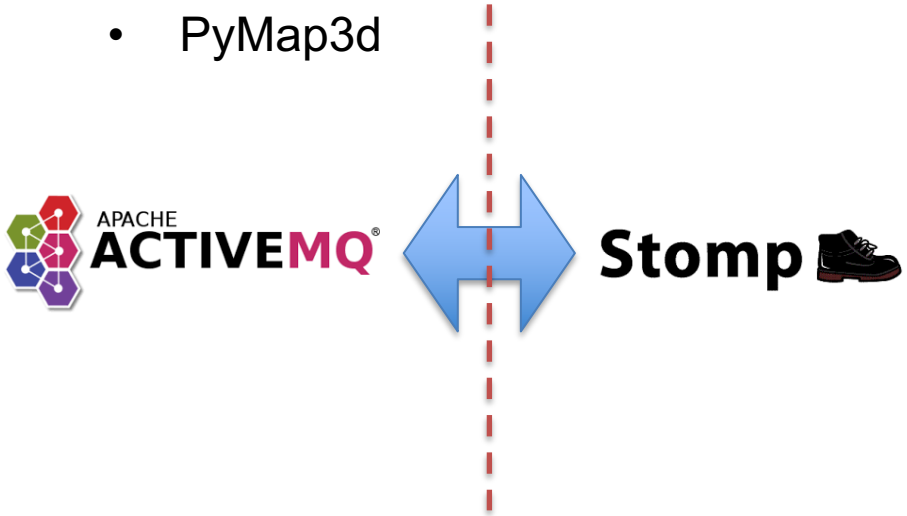
Language
- Python

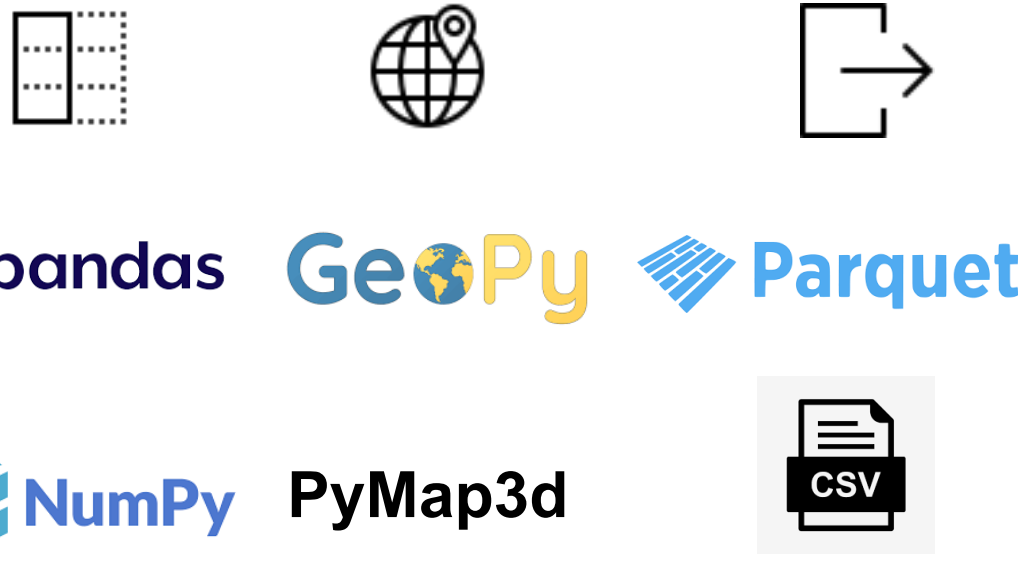
Data Manipulation
- Pandas
- Numpy

Geospatial Data Handling
- GeoPy
- PyMap3d

Data Output Format
- Parquet
- CSV

Interact with ActiveMQ Messages
- STOMP (protocol and python library)





IDP Data Flow

Code

DGPS\20210312\raw

NC1-DT1-BuildupRun2--DGPS--Sortie_1--20210312.csv

APACHE ACTIVEMQ

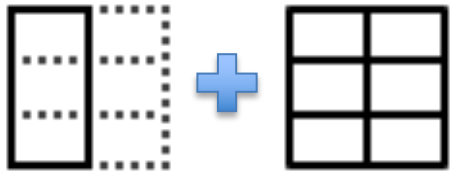
idp_messenger.py

process_idp.py

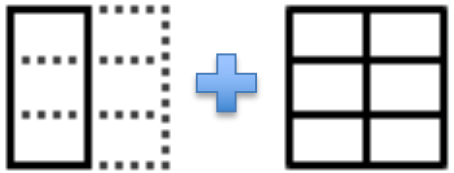
```
PROCESSORS = {
  'ias': idp_processors.process_ias,
  'rr': idp_processors.process_rr,
  'wisk': idp_processors.process_wisk,
  'wisk_dgps': idp_processors.process_wisk_dgps,
  'wisk_surface_weather': idp_processors.process_wisk_surface_weather,
  'wisk_weather': idp_processors.process_wisk_weather,
  'joby': idp_processors.process_joby,
  'iads': idp_processors.process_iads,
  'dgps': idp_processors.process_dgps,
  'sbsm': idp_processors.process_sbsm,
  'adsb': idp_processors.process_adsb,
  'fiapa': idp_processors.process_fiapa,
  'weather_sodar': idp_processors.process_sodar,
  'weather_surface_station': idp_processors.process_surfaceWx,
  'distances': idp_processors.process_distances
}
```

Data

Time-Series Only
- Fixed intervals according to config



Insert New Data (Processed)
- Merge on timestamps
- **Yields last known value from new dataset (forward filling)**



Repeat for each data source

	idp_utc_time	ias_bssrolldeg	ias_bsspitchdeg	ias_bstrueheadingdeg
72886	2023-06-27 16:45:25.500000+00:00	12.767732	3.498820	281.311401
33109	2023-06-27 16:12:16.650000+00:00	0.077970	2.156518	91.859535
29026	2023-06-27 16:08:52.500000+00:00	-0.024397	1.145478	171.397247
22050	2023-06-27 16:03:03.700000+00:00	-0.576566	2.484171	102.739548
24357	2023-06-27 16:04:59.050000+00:00	3.841762	3.219630	79.937714

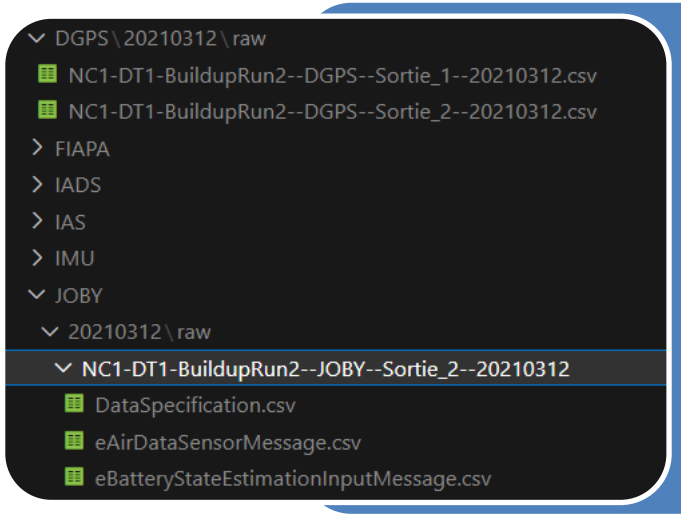
	synchronized_utc	roll_bss	pitch_bss	true_heading_bss	pitch_rate_bss	roll_rate_bss
36240	2023-06-27 16:14:53.200000+00:00	25.799538	1.695812	195.005524	2.311327	-0.554954
57756	2023-06-27 16:32:49+00:00	-1.088634	6.017719	208.707825	0.180103	-0.652093
112594	2023-06-27 17:18:30.900000+00:00	13.208816	1.603979	75.139229	0.753909	-2.573450
16513	2023-06-27 15:58:26.850000+00:00	1.446432	-0.175781	139.449417	-0.015381	-0.187245
106278	2023-06-27 17:13:15.100000+00:00	14.167637	2.256191	229.014053	1.073866	-3.661975

IDP Design Pattern / Under the Hood

Process of extending IDP to handle new event involves defining or modifying:

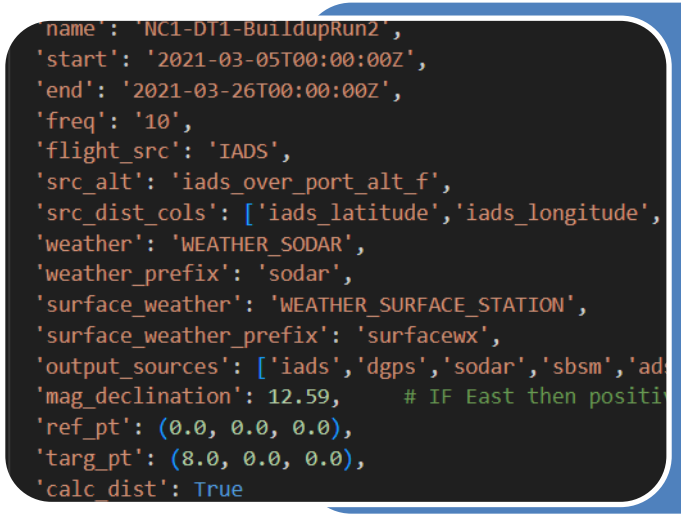
- File naming convention
- Config File
- IDP Processors
- IDP Utils
- State Tracker / File
- Internal to External Mapper (field names and units of measure)

IDP Design Pattern / Under the Hood



File Name Convention

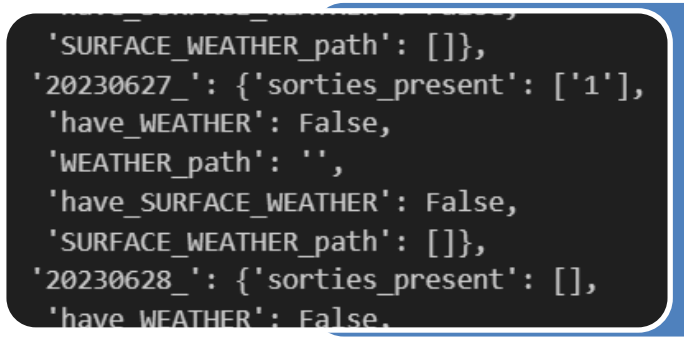
- Event--Source--Sortie_N--YYYYMMDD
- File name OR Folder name



Event Config

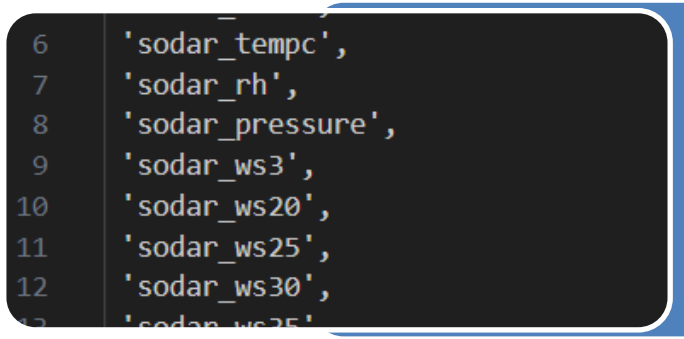
- Simple JSON / Key-Value Object
- In a .py file

IDP Design Pattern / Under the Hood



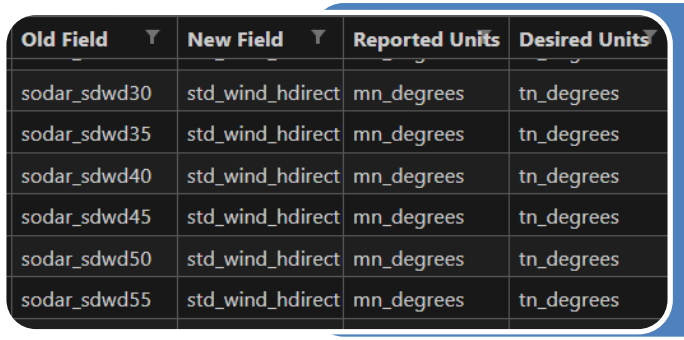
State Tracker

- JSON file for each Event
- Tracks state for each day of event
- Sorties, Weather, etc…



Columns File

- Simple .txt file with list of all column names across all data sources
- Currently over 1100 items



Old Field	New Field	Reported Units	Desired Units
sodar_sdwd30	std_wind_hdirect	mn_degrees	tn_degrees
sodar_sdwd35	std_wind_hdirect	mn_degrees	tn_degrees
sodar_sdwd40	std_wind_hdirect	mn_degrees	tn_degrees
sodar_sdwd45	std_wind_hdirect	mn_degrees	tn_degrees
sodar_sdwd50	std_wind_hdirect	mn_degrees	tn_degrees
sodar_sdwd55	std_wind_hdirect	mn_degrees	tn_degrees

Mapping File

- Simple .csv file
- Internal name and unit >> Output name and unit

IDP Design Pattern / Under the Hood

```
Clean and prep Reliable Robotics data. Save clean version.
Insert clean version to IDP file or table.
'''
df = idp_utils.prep_rr(path, ts_df)

# save cleaned data source
idp_utils.idp_save(df, src.lower(), date, sortie, icao)

# trim ts_df to df time_range
start = pd.to_datetime(df.head(1).rr_idp_utc_time.values[0], utc=True)
end = pd.to_datetime(df.tail(1).rr_idp_utc_time.values[0], utc=True)
ts_df = ts_df.query('(idp_utc_time >= @start) and (idp_utc_time <= @end)')

# Merge onto event timeseries and backfill
df = pd.merge_asof(
  ts_df,
  df,
  left_on='idp_utc_time',
  right_on='rr_idp_utc_time',
  direction='backward'
)\
  .drop(columns=['rr_idp_utc_time'])
```

IDP Processor Function

- Processes new data source
- Merges onto IDP time-series
- Sometimes performs additional tasks depending on Data Source
- Returns result to main process_idp.py process for insert into IDP

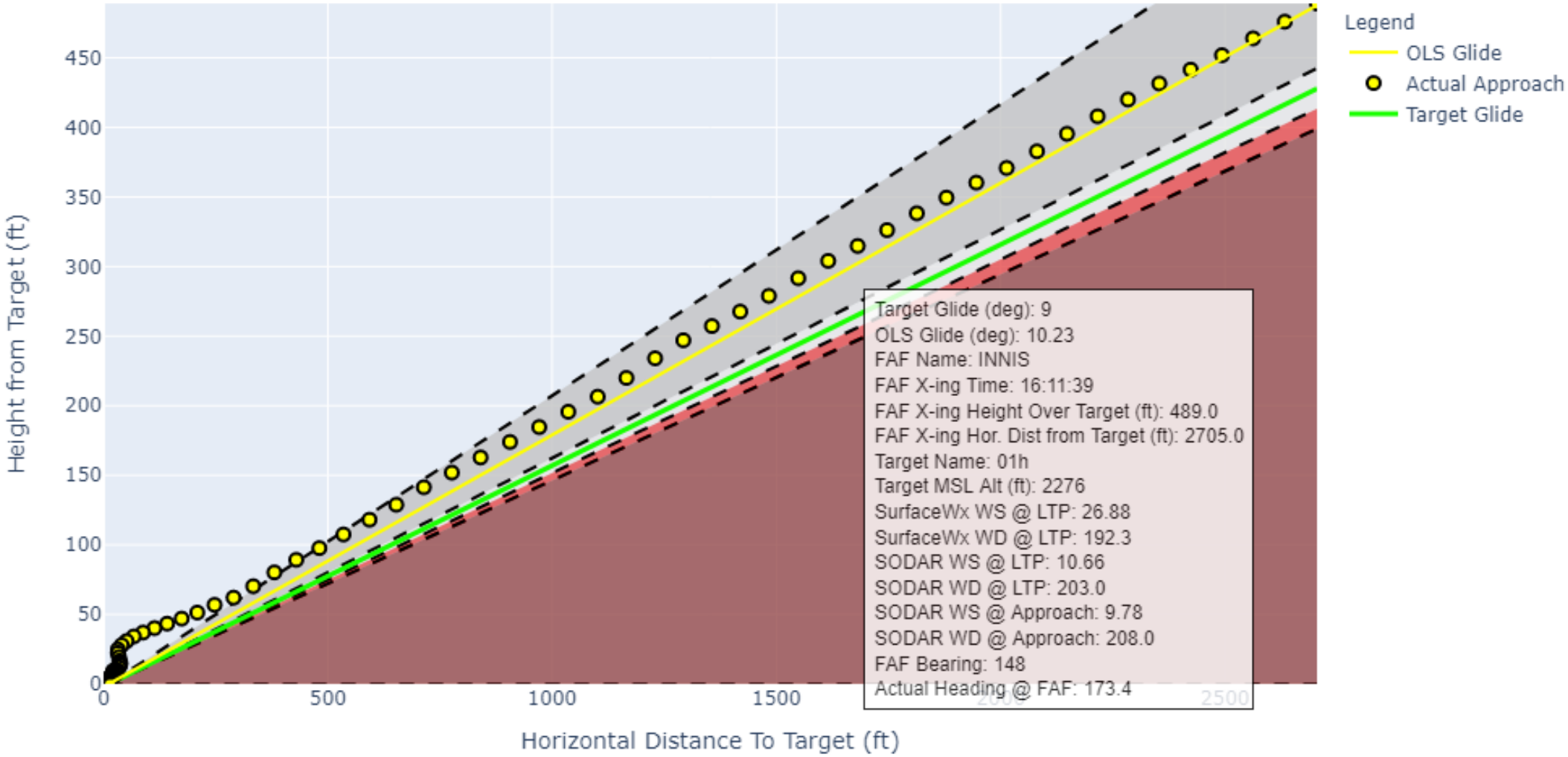
```
prep_dgps(path):
'''
path (str): relative path to raw data file

return (Pandas DF): cleaned DF
'''
printlog('\n\tdGPS data read from {}'.format(path))

utc_date_cols = ['Lo','T','Lo.1','T.1','Lo.2']
utc_time_cols = ['GP','T.2','GP.1','T.3','GPSTime']
dgps_drop_cols = utc_date_cols + utc_time_cols + ['GPSTime.1','GPS_D','GPS_T']
dgps_m_to_f_cols = ['H_Ell','VEast','VNorth','VUp']
dgps_in_feet_cols = [c+'_f' for c in dgps_m_to_f_cols]
dgps_in_meters_cols = [c+'_m' for c in dgps_m_to_f_cols]

dgps_df = pd.read_csv(path, skiprows=5, delim_whitespace=True)\
    .assign(GPS_D=lambda df: df[utc_date_cols].apply(concat_cols, axis=1))\
    .assign(GPS_T=lambda df: df[utc_time_cols].apply(concat_cols, axis=1))\
    .assign(GPS_Time=lambda df: df.GPS_D + ' ' + df.GPS_T)\
    .drop(columns=dgps_drop_cols)\
    .assign(GPS_Time=lambda df: pd.to_datetime(df['GPS_Time']))\
    .rename(columns={
      'UTC-Corr':'UTCCorr',
      'H-Ell':'H_Ell'
    })\
    .assign(UTC_Time=lambda df: df.UTCDate + ' ' + df.UTCCorr)\
```

IDP Utils

- Contains custom prep_datasource functions for each data source
- Primary transformations on Data Source
- Returns results to idp_processor

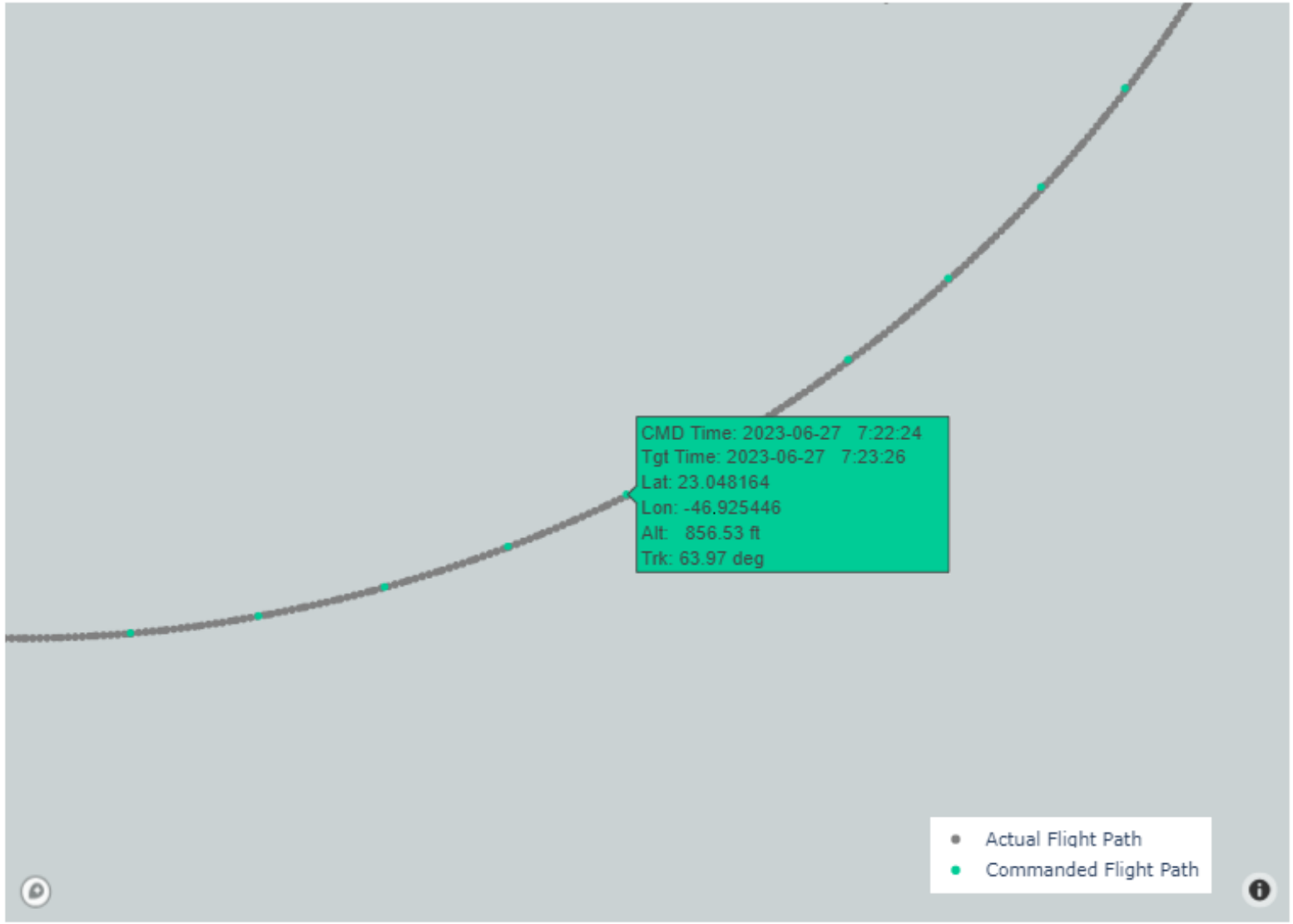
IDP Applications

Glide Path Analysis: How well can an aircraft conform to an approach path?



IDP Applications cont.

Charting 4DT Performance: How well can an aircraft conform to a four-dimensional trajectory



IDP Try it at Home

- Create a notional IDP at home using Python
- Code combines three csv files and creates a 2D chart
    1. Aircraft 1 (AC1.csv)
    2. Aircraft 2 (AC2.csv)
    3. Wind data (Wind.csv)
- Required Python packages:
    - Pandas
    - Plotly.express
- Four functions:
    1. main
    2. create_initial_df
    3. create_idp
    4. plot_flight_path

IDP Try it at Home

AC1.csv
- 30 minutes of data
- 3 fields
- 10 Hz

utc_timestamp	latitude_deg	longitude_deg	altitude_msl
2023-06-06 20:30:00.040000+00:00	32.33901982	-110.9581413	5715.138177
2023-06-06 20:30:00.140000+00:00	32.33906042	-110.9581073	5715.542063
2023-06-06 20:30:00.240000+00:00	32.33910101	-110.9580733	5714.797188
…	…	…	…
2023-06-06 20:59:59.980000+00:00	32.31180985	-110.9370047	5658.16578

AC2.csv
- 30 minutes of data
- 3 fields
- 10 Hz

utc_timestamp	latitude_deg	longitude_deg	altitude_msl
2023-06-06 20:30:00.070000+00:00	32.16469516	-110.9078251	6265.111541
2023-06-06 20:30:00.170000+00:00	32.16464474	-110.9077954	6264.688974
2023-06-06 20:30:00.270000+00:00	32.16459433	-110.9077657	6264.843051
…	…	…	…
2023-06-06 20:59:59.960000+00:00	32.226841	-110.8565603	6800.424418

Wind.csv
- 30 minutes of data
- 2 fields
- 1/60 Hz

utc_timestamp	wind	windgust
2023-06-06 20:30.000000+00:00	5.865259	11.98399
2023-06-06 20:31.000000+00:00	4.255078	8.361136
2023-06-06 20:32.000000+00:00	5.247163	11.53886
…	…	…
2023-06-06 21:00.000000+00:00	5.447292	11.02237

IDP Try it at Home cont.

1. main

```
# the main entry point. Define start, send, idp time freq, input files, and output file name
if __name__ == '__main__':
    start_time = "2023-06-06 20:30.000000+00:00" # the starting timestamp of your idp
    end_time = "2023-06-06 21:00.000000+00:00" # the ending timestamp of your idp
    idp_time_freq_hz = 10 # the number of timestamps per second of your idp
    input_files = ["AC1", "AC2", "Wind"] # your source input files

    idp = create_idp(start_time, end_time, idp_time_freq_hz, input_files) # create an idp dataframe
    idp.to_csv("IDP_example.csv")  # save the dataframe as a csv

    plot_flight_path(idp) # create an interactive 2D chart
```

IDP Try it at Home cont.

2. create_initial_df

```
# Creates the initial dataframe with a single time field ranging from start to end at the specified frequency
def create_initial_df(start_time, end_time, idp_time_freq_hz):
    seconds = (pd.to_datetime(end_time) - pd.to_datetime(start_time)).total_seconds() # number of total seconds
    periods = int(idp_time_freq_hz) * seconds # number of periods to generate
    freq = '{}ms'.format(int(1000 / int(idp_time_freq_hz))) # idp freq

    dt_df = pd.DataFrame()
    dt_df['utc_timestamp'] = pd.date_range(start_time, periods=periods, freq=freq)

    return dt_df
```

IDP Try it at Home cont.

3. create_idp

```
# Creates the idp dataframe from merging input files into the initial dataframe
def create_idp(start_time, end_time, idp_time_freq_hz, input_files):
    idp_df = create_initial_df(start_time, end_time, idp_time_freq_hz) #create the initial dataframe
    for in_file in input_files: # read all input files and merge with idp_df
        temp_df = pd.read_csv(in_file + ".csv") # read input file as temporary dataframe
        temp_df["utc_timestamp"] = pd.to_datetime(temp_df["utc_timestamp"]) # convert timestamp to proper datetime
        # append filename to column name
        temp_df = temp_df.rename(columns={c: c + "_" + in_file for c in temp_df.columns if c not in ['utc_timestamp']})
        temp_df.sort_values(by='utc_timestamp', ascending=True, inplace=True)  # sort the input file prior to merging
        # merge on utc_timestamp using a backward direction
        idp_df = pd.merge_asof(idp_df, temp_df, on="utc_timestamp", direction='backward')

    return idp_df
```

IDP Try it at Home cont.

4. plot_flight_path

```
# create 2D flight path map of AC1 and AC2 with wind data in hover info
def plot_flight_path(idp_df):
    fig = px.scatter_mapbox(idp_df, lat="latitude_deg_AC1", lon="longitude_deg_AC1", hover_name="utc_timestamp",
                            hover_data=["latitude_deg_AC1", "longitude_deg_AC1", "altitude_msl_AC1", "wind_Wind",
                                        "windgust_Wind"],
                            color_discrete_sequence=["fuchsia"], zoom=10, height=600)

    fig2 = px.scatter_mapbox(idp_df, lat="latitude_deg_AC2", lon="longitude_deg_AC2", hover_name="utc_timestamp",
                            hover_data=["latitude_deg_AC2", "longitude_deg_AC2", "altitude_msl_AC2", "wind_Wind",
                                        "windgust_Wind"],
                            color_discrete_sequence=["blue"], zoom=10, height=600)
    fig.add_trace(fig2.data[0])

    fig.update_layout(
        mapbox_style="white-bg",
        mapbox_layers=[
            {
                "below": 'traces',
                "sourcetype": "raster",
                "sourceattribution": "United States Geological Survey",
                "source": ["https://basemap.nationalmap.gov/arcgis/rest/services/USGSImageryOnly/MapServer/tile/{z}/{y}/{x}"]
            }
        ])
    fig.update_layout(margin={"r": 0, "t": 0, "l": 0, "b": 0})
    fig.show()
```

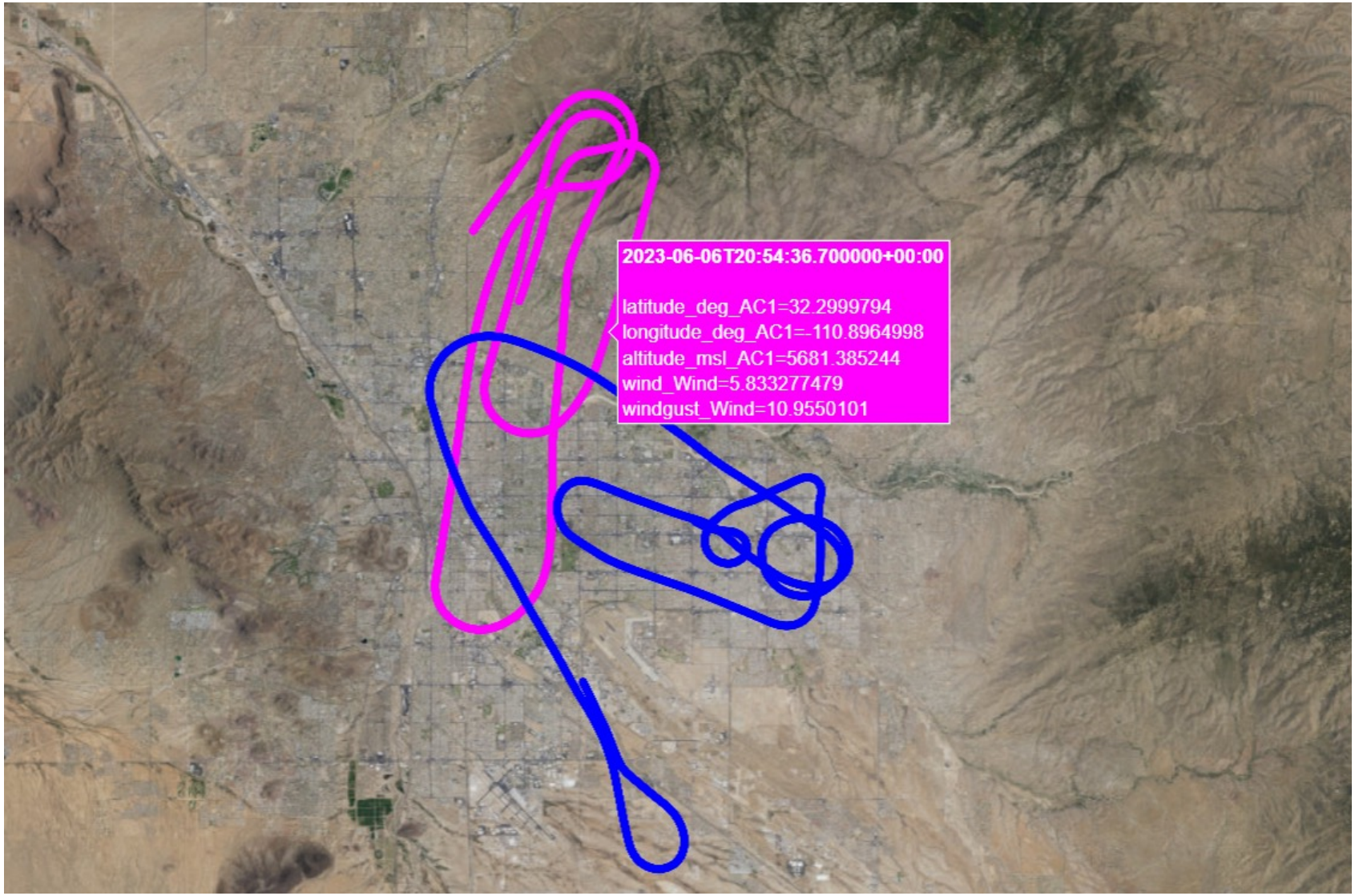
IDP Try it at Home cont.

Produced IDP

utc_timestamp	latitude_deg_AC1	longitude_deg_AC1	altitude_msl_AC1	latitude_deg_AC2	longitude_deg_AC2	altitude_msl_AC2	wind_Wind	windgust_Wind
2023-06-06 20:30:00+00:00	32.33901982	-110.9581413	5715.138177	32.16469516	-110.9078251	6265.111541	5.865258982	11.98399338
2023-06-06 20:30:00.100000+00:00	32.33906042	-110.9581073	5715.542063	32.16469516	-110.9078251	6265.111541	5.865258982	11.98399338
2023-06-06 20:30:00.200000+00:00	32.33910101	-110.9580733	5714.797188	32.16464474	-110.9077954	6264.688974	5.865258982	11.98399338
2023-06-06 20:30:00.300000+00:00	32.3391416	-110.9580393	5714.953099	32.16459433	-110.9077657	6264.843051	5.865258982	11.98399338
2023-06-06 20:30:00.400000+00:00	32.33918218	-110.9580053	5715.281096	32.16454392	-110.907736	6264.740423	5.865258982	11.98399338
2023-06-06 20:30:00.500000+00:00	32.33922275	-110.9579713	5714.885721	32.16449351	-110.9077063	6264.538827	5.865258982	11.98399338
2023-06-06 20:30:00.600000+00:00	32.33926331	-110.9579372	5715.67714	32.1644431	-110.9076766	6265.179295	5.865258982	11.98399338
2023-06-06 20:30:00.700000+00:00	32.33930387	-110.9579032	5714.786375	32.16439269	-110.9076469	6265.119557	5.865258982	11.98399338
...	...	...	...	...	...	...	...	...
2023-06-06 20:59:59.400000+00:00	32.31209516	-110.9368824	5658.881337	32.22668636	-110.8562657	6803.44554	5.447292368	11.02236899
2023-06-06 20:59:59.500000+00:00	32.3120476	-110.9369028	5658.095025	32.22671204	-110.8563148	6802.779681	5.447292368	11.02236899
2023-06-06 20:59:59.600000+00:00	32.31200005	-110.9369233	5658.19658	32.2267376	-110.856364	6802.846551	5.447292368	11.02236899
2023-06-06 20:59:59.700000+00:00	32.31195251	-110.9369437	5658.534737	32.22676357	-110.8564133	6801.820085	5.447292368	11.02236899
2023-06-06 20:59:59.800000+00:00	32.31190496	-110.936964	5658.82289	32.22678926	-110.8564624	6801.421869	5.447292368	11.02236899
2023-06-06 20:59:59.900000+00:00	32.31185741	-110.9369844	5658.417046	32.22681518	-110.8565113	6800.932537	5.447292368	11.02236899

IDP Try it at Home cont.

Produced Chart



IDP Questions

?