AiTR STANDARDS: COMMON DATA FORMAT

PREPARED BY:
U.S. Army Combat Capabilities Development Command (DEVCOM)
C5ISR Center
Aberdeen Proving Ground, MD 21005-1845

Version: 1.0
Release Date: 1 September 2023

Distribution Statement A: Approved for public release. Distribution is unlimited.

Distribution Statement A: Approved for public release. Distribution is unlimited.

NOTICES

Disclaimer

The findings in this report are not to be construed as an official Department of the Army position unless so designated by other authorized documents.

Citation of manufacturer or trade names does not constitute an official endorsement or approval of the use thereof.

Distribution Statement A: Approved for public release. Distribution is unlimited.

Distribution Statement A: Approved for public release. Distribution is unlimited.

This page is intentionally left blank

Distribution Statement A: Approved for public release. Distribution is unlimited.

Distribution Statement A: Approved for public release. Distribution is unlimited.

This page is intentionally left blank

Distribution Statement A: Approved for public release. Distribution is unlimited.

Foreward

Aided/Automatic Target Recognition (AiTR/ATR), referred to as AiTR in this document, describes a broad range of automated or assisted exploitation functions, including automated and semi-automated detection, false alarm mitigation, classification, recognition, and identification[1]. AiTR has been in development since the 1970s but has recently seen a renewed interest due to advances in commercially available high-performance processing devices, such as Graphics Processing Units (GPUs). An ongoing challenge with AiTR systems is the dependency on vast amounts of labeled training data. The importance of high-quality training data on targets of interest cannot be overstated. It is a key factor in the success or failure of AiTR algorithms.

In an effort to maximize efficiencies in AiTR development and sustainment, a collaborative group from C5ISR Center, Army Research Lab (ARL), and the Artificial Intelligence Integration Center (AI2C) developed a set of common standards to enable sharing of AiTR training data across both the Army enterprise and industry partners. These standards are a collection of three documents, each addressing a critical element of AiTR data that must be synchronized to effectively share AiTR training data. The first document is the Common Data Format (CDF). The CDF specifies a single file structure that is capable of representing imagery, sensor metadata, terrain and target data, labels, and other relevant information. The second document creates a set of common labeling guidelines. Consistency in training data labels is a critical factor in sharing data between AiTR programs. This document establishes a set of best practices for ensuring consistent labels and defines norms for addressing various edge cases. The entitled Target Acquisition Ontology (TAO) is the third document covered under the AiTR Standards. The TAO is a consolidated ontology that standardizes target terminology for sharing of imagery and data across programs. The current TAO focuses on ground-based targets but is extensible to additional targets.

To maintain and evolve these standards, a change board has been established to review and adjudicate any requests for change to the documents on a periodic basis. To request a change or to send comments/questions to the board, please e-mail ***usarmy.belvoir.devcom-c5isr.mbx.aitr-standards@army.mil***. Make sure to include the specific document and page number relevant to the request and clearly explain the requested modification with supporting justification.

[1] Definition published in 2002 by Deputy Under Secretary of Defense for Science and Technology Sensor Systems (DUSD (S&T/SS)), ATR Working Group (ATRWG) Ad-hoc Committee on Problem Sets

Distribution Statement A: Approved for public release. Distribution is unlimited.

Contents

List of Tables

List of Figures

Distribution Statement A: Approved for public release. Distribution is unlimited.

Overview

A typical aided target recognition (AiTR) data collection is comprised of many different data modalities and formats (e.g., imagery from different cameras, Global Positioning System (GPS), weather stations, black body). Easy access to these data attributes is valuable to both algorithm training and evaluation processes. This data is often scattered and requires additional steps to parse and extract. Additionally, no consensus exists within the community on data format standards and often times a given organization will use multiple different formats due to differing program requirements (e.g., ARF, UBIN, MSF, ENVI, AGT, DVT). Most of these formats convey the same information (e.g., imagery, bounding box, and minimal metadata); however, most currently used formats are not inter-operable.

The purpose of this document is to identify a standard file format and structure to enable fast data sharing for algorithm training and evaluation that would be flexible and scalable for use by other DoD organizations. The Common Data Format (CDF) allows for a centralized, standardized, and streamlined method of storing AiTR related data. This allows for easy sharing and discovery of AiTR data. The following section details the CDF format, and its usage as suggested by the authors of this document.

Distribution Statement A: Approved for public release. Distribution is unlimited.

1. Introduction

Hierarchical Data Format 5 (HDF5) was chosen to be the basis for a common data format. HDF5 is capable of storing an arbitrary number of files in mixed format and precision. For AiTR data collections, this means one could store imagery from sensors along with target labels and metadata into a single HDF5 file. An AiTR dataset would then be comprised of a series of HDF5 files (i.e., one per scenario) where all data and metadata sub-files within an HDF5 file would be synchronized. Synchronizing and aggregating related sensor data into a discrete collection with a common metadata structure and ontology satisfies the requirement for a common format. Furthermore, it would make the training data highly modular and portable.

To further define this common data format, desired metadata was identified, and a structure was proposed that could comprise the future HDF5 file format. This format is now referred to as the CDF throughout this document.

2. Data Format Summary

The HDF5 file type was chosen for this effort due to its ability to organize and store collected scene imagery and all relevant metadata into a single, well-structured package. Each HDF5 file contains the raw recorded imagery data from a single camera/sensor during a collected scene, defined as a sequence of frames from a continuous video, along with all the relevant metadata. The goal is to encapsulate all collected data for the particular scene into a single file package which can easily be shared with other organizations and utilized for AiTR training purposes. HDF5 group attributes are utilized for surface-level information about the scene, while much of the metadata is stored using standardized JavaScript Object Notations (JSON) formats. The following sections will discuss how both the typical HDF5 storage is used, along with a more in-depth look at how the JSON format is employed for storing metadata.

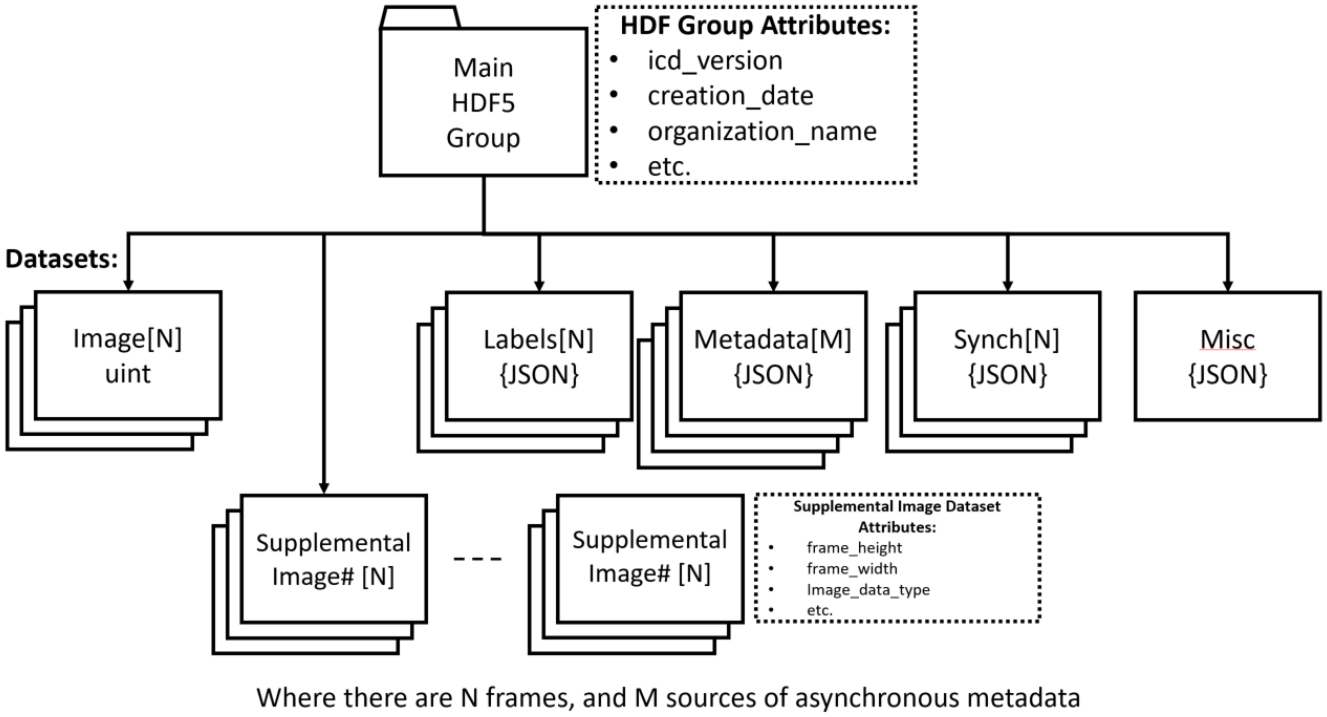
2.1 Data Format

The organizational structure of a HDF5 file is similar to the way files and subdirectories are organized within a directory of a file system. Units of data, such as an image stack or metadata dictionary, are placed into "datasets". Datasets can be further combined into units called "groups" which are like the subdirectories. Both datasets and groups can be assigned properties called "attributes" which provide additional context and information. HDF5 attributes allow for any type of metadata to be attached to its groups and datasets. For example, the HDF5 file format has attributes that specify where and when the data was collected and by what type of camera/sensor. The format described in this guide only utilizes a single group (the main HDF5/root group) along with a few standard datasets. Within the HDF5 file datasets are available to be stored: the sequence of raw image data from a recorded scene, the sequence of object/target labels JSONs for each frame, the set of metadata source JSONs, the sequence of JSONS where metadata has been synchronized for each frame, and a miscellaneous dataset for

Distribution Statement A: Approved for public release. Distribution is unlimited.

storing extra data that does not fit the standard. In addition, any number of supplemental image datasets that contain image-type data relevant to the image dataset may exist. As per HDF5 data structure, each dataset must contain the same data type. **Error! Reference source not found.** is a visual representation of the common data format:

Figure 2.1-1: Common Data Format Architecture



(*Note that alarm files are not included in this format and are specified by a separate format. Please refer to section 4 for information regarding alarm files.)*

3. Data Format Detailed Description

The HDF5 format outlined in this guide only utilizes the attributes of the main/root HDF5 group. The HDF5 group attributes store the high-level metadata in such a way where it is readily accessible to HDF5 file viewers and other software using the HDF5 Application Programming Interface (API). The imagery of single scene collection is stored into a typical HDF5 dataset which is a large array of numeric values. The remainder of the metadata associated with the collected scene needs to be added to the same HDF5 file. This is where the CDF diverges from normal HDF5 use-cases: when storing data in the label, metadata, and synchronous datasets. These three datasets stored serialized JSONs containing all the metadata

Distribution Statement A: Approved for public release. Distribution is unlimited.

which is relevant to individual frames and objects in the scene. This design was selected so that frame level data could easily be extracted from the HDF5 file without any intermediate processing step.

The following sections will describe how each of these datasets are used, how the JSONs should be formatted, and what types of keys and values are allowable in the format. Generally, key-value pairs are used for storing information in both the HDF5 group attributes and JSON format. The key is a unique identifier which is used to label and look up a corresponding unit of stored data. For example, “schema_guide_version”: “1.0.0.” The unique key “schema_guide_version” corresponds to a value “1.0.0.” This guide gives allowable keys and corresponding value types supported by our CDF API.

3.1 HDF5 Group Attributes

The HDF5 group attributes are utilized to store high level metadata and information about the scene that does not vary throughout the duration of the scene collection. This includes information such as the what, when, and where the scene was captured along with information about the sensor used to collect the video/frames. Not all HDF5 key-value pair fields are guaranteed to have a value since some information may not be collected or is not relevant, depending on the collected scene. **Error! Reference source not found.** lists the standardized, possible HDF5 attribute keys, its format, and a description of the information stored. While not encouraged, CDF users also have the option to add custom key-value pairs via configuration files provided in the API. Please refer to the API documentation for further information in the “readme” files located in the source code.

***Table 3.1-1:** CDF Group Attributes*

Fields highlighted in green are mandatory for CDF compatibility

HDF Key	HDF Value Format	Description
organization_name	string	Name of the organization
division_name	string	Division or lab name
branch_name	string	Name of branch
project_type	string	Global ID intended to uniquely identify all files associated with a project
project_name	string	Name of project
collection_name	string	Unique name for the collection (e.g. a combination of project name, collection site, collection time period)
type_of_data	string	Type of data (e.g., real or synthetic)
site_identifier	string	Unique ID of collection site
collection_datetime_UTC	string	Start date and time of collection in UTC (YYYY-MM-DDThh:mm:ss.ss+00:00)
collection_datetime_local	string	Start date and time of collection in local time (YYYY-MM-DDThh:mm:ss.ss+hh:mm)
collection_time_zone	string	Time zone of collection site (e.g., U.S./Eastern or U.S./Pacific)

Distribution Statement A: Approved for public release. Distribution is unlimited.

HDF Key	HDF Value Format	Description
collection_time_UTC_offset	string	UTC time offset hh:mm
scene_number	integer	Number of scene from the collection plan
scene_look_number	integer	Number of look in case of repeat collections of the scene
dataset_POC	string	Contact information for POC familiar with the dataset/collection
object_class_list	array of string	Array of all object classes that appear in at least one frame of the scene
clutter_list	array of string	Array of all clutters that appear in at least one frame of the scene
schema_guide_version	string	CDF Schema Guide version used for creating the file
ontology	string	Ontology name, version, base Internationalized Resource Identifier (IRI) used for assigning object classes
hdf5_creation_datetime_UTC	string	Creation time of HDF5 in UTC (YYYY-MM-DDThh:mm:ss.ss+00:00)
hdf5_creation_datetime_local	string	Creation time of HDF5 in local time (YYYY-MM-DDThh:mm:ss.ss+hh:mm)
hdf5_modified_datetime_UTC	string	Modification time of HDF5 in UTC (YYYY-MM-DDThh:mm:ss.ss+00:00)
hdf5_modified_datetime_local	string	Modification time of HDF5 in local time (YYYY-MM-DDThh:mm:ss.ss+hh:mm)
hdf5_phase	integer	Current phase information of the HDF5 file
config_file_path	string	Path to the customized CDF config file
labeling_guidelines	string	Link to an external document describing the standard labeling guidelines used
preprocessing_guidelines	string	Description of pre-processing guidelines applied to the entire image dataset (e.g., "used camera sim to add sensor noise")
data_generation_guidelines	string	Guidelines on data creation (e.g., target location radius)
distribution_statement	string	CDF distribution statement guide (e.g. Distribution Statement A: Approved for public release. Distribution is unlimited.)
is_CDF_compliant	bool	A boolean specifying if the file is CDF compliant. This is automatically populated by the CDF tools
frame_height	integer	Height of the frame in pixels (px)
frame_width	integer	Width of the frame in px
number_of_channels	integer	Number of channels per frame
channel_type	string	Description of channel storage scheme
pixel_format_name	string	Format type of the image pixels (e.g. Mono8. Refer to GenICamPixelFormats Document)
number_of_frames	integer	Number of frames
frame_rate	integer	Frame rate of camera in frames per second (fps)
preprocessing	string	Description of pre-processing performed on the imagery
sensor_name	string	Name of the sensor

Distribution Statement A: Approved for public release. Distribution is unlimited.

HDF Key	HDF Value Format	Description
sensor_specs	string	Additional sensor specs (e.g., DiaphragmBladeCount)
sensor_identifier	string	Unique ID for sensor
sensor_model	string	Model of the sensor
sensor_type	string	Type of sensor (e.g., visible, midwave infrared [MWIR], longwave infrared [LWIR])
sensor_subtype	string	Subtype of sensor (e.g., uncooled, cooled)
sensor_location_latitude	float	Sensor latitude at collection start in degrees (deg)
sensor_location_longitude	float	Sensor longitude at collection start in deg
sensor_relative_coordinates	float	Sensor relative coordinates (x, y, z) to the reference point (m)
sensor_relative_rotation	array of float	Sensor relative rotation (pitch, yaw, roll) to the reference coordinate (deg)
integration_time	array of float	Integration time of sensor in milliseconds (ms)
gain_level	string	Qualitative measure of gain (e.g., high or low)
gain	array of float	Quantitative measure of gain
sensor_known_latency	integer	Sensor latency in milliseconds (ms)
gamma_of_imagery	float	Gamma value for viewing imagery
mtf_collection_datetime_UTC	string	Date/time of Modulation Transfer Function (MTF) collection (YYYY-MM-DDThh:mm:ss.ss+00:00)
mtf_of_sensor	array of float	2D array of values describing the radially symmetric MTF of the sensor
psf_of_sensor	array of float	2D array of values describing the radially symmetric Point Spread Function (PSF) of the sensor
sensor_pan_rate	array of float	Sensor pan rate in degrees per second (deg/s)
NUC_datetime_UTC	string	Time of last sensor Non-Uniformity Correction (NUC) (YYYY-MM-DDThh:mm:ss.ss+00:00)
NUC_type	string	Type of NUC (e.g., 1, 2, multi-point)
focusing_type_of_camera	string	Type of camera focusing (e.g., manual or auto)
sensor_focal_lengths	array of float	Sensor calibrated/effective focal length (mm)
sensor_f_numbers	array of float	F# of sensor
sensor_avg_h_fov	float	The average horizontal field of view of the sensor (deg)
sensor_avg_v_fov	float	The average vertical field of view of the sensor (deg)
intensity_to_temp_map	array of float	Array describing mapping of pixel intensity to temperature in Kelvin (K)
well_capacity	float	Well capacity in electrons (e-)
dark_current_density	array of float	Dark current density in nanoAmps/cm^2
total_downstream_noise	float	Total downstream noise in Root-Mean-Squared (RMS) electrons (e-)

Distribution Statement A: Approved for public release. Distribution is unlimited.

HDF Key	HDF Value Format	Description
spectral_quantum_efficiency	array of float	2xN array describing spectral quantum efficiency (QE) where first row is wavelength in micrometers (μm) and second row is dimensionless efficiency value between 0 and 1
spectral_quantum_efficiency_mean	float	Sensor QE mean
spectral_quantum_efficiency_stdev	float	Sensor QE standard deviation
horizontal_detector_dimension	float	Horizontal detector dimension in μm
vertical_detector_dimension	float	Vertical detector dimension in μm
horizontal_detector_count	integer	Horizontal detector count
vertical_detector_count	integer	Vertical detector count
noise_equivalent _temperature_difference	float	Noise equivalent temperature difference in Kelvin (K)
cut_on_wavelength	float	Cut-on wavelength in μm
cut_off_wavelength	float	Cut-off wavelength in μm
is_sensor_platform_moving	bool	Is the sensor on a moving platform
minimum_collected_object_range	float	The minimum range between object and the collection sensor (meters [m])
maximum_collected_object_range	float	The maximum range between object and the collection sensor (m)
minimum_platform_altitude	float	The minimum altitude of collection platform (m)
maximum_platform_altitude	float	The maximum altitude of collection platform (m)
percent_labeled	float	Percentage of labeled frames (in the scene)
percent_quality_controlled	float	Percentage of quality controlled labeled frames (in the scene)
labeling_POC	array of strings	A list of labeling point of contact (POC) names
labeling_history	array of strings	Array of strings holding labeling history such as labeling organization name and labeling efforts date
labeling_notes	string	Additional information about label
excluded_frame_labeling_notes	array of strings	Array of strings with reasoning behind the excluded_frames entries
excluded_frame_scoring_notes	array of strings	Array of strings with reasoning behind the excluded_frames_scoring entries
collection_unique_object_ID	array of strings	Array of strings containing object IDs unique throughout the entire collection
scene_unique_object_ID	array of strings	Array of strings containing object IDs unique throughout the entire scene
non_unique_object_ID	array of strings	Array of strings containing object IDs non-unique throughout the entire scene
extraneous_object_class_list	array of strings	A list of possible low fidelity objects, present in the data but not well labeled in the dataset

Distribution Statement A: Approved for public release. Distribution is unlimited.

HDF Key	HDF Value Format	Description
collection_object_class_list	array of strings	An object class list of entire collection (as opposed to object_class_list that applies to the scene only)
color_correction_matrix	array of floats	The matrix used for color correction in the main image dataset
color_correction_offset	array of floats	Offset values used for color correction in the main image dataset

3.2 Image and Supplemental Image Datasets

The image dataset stores the image data for the video in a large array of integers that is shaped according to the pixel dimensions of the individual frame, the number of frames, and the number of channels. GenICam Pixel Formats document[1] outlines the various types of image pixel formats that can be stored along with short descriptions of each. Please note that the CDF visualization tools, by default, will support standard red, green, blue (sRGB) for colored and linear monochrome for gray scale (e.g. infrared (IR)) images. Users are welcome to use other formats (e.g. Bayer), however the current visualizer will not be able to display that image.

The dataset will be shaped as follows: $frames \times height \times width \times channels$. For example, 100 frames of monochrome 16-bit video that is 1280x720 would be a 'uint16' dataset with the dimensions $100 \times 720 \times 1280 \times 1$. In the case of 100 frames of 1920x1080 RGB24 imagery where there are three 8-bit channels, the dataset would be a 'uint8' dataset with dimensions $100 \times 1080 \times 1920 \times 3$. In the case of 100 frames of 640x320 hyperspectral imagery with N 8-bit channels, the dataset would be a 'uint8' dataset with dimensions $100 \times 320 \times 640 \times \mathrm{N}$.

Sub-image datasets could hold image-type data such as object masks, object range map, scene temperature maps, etc. These datasets have the same general structure as the image dataset and are shaped as $frames \times height \times width \times channels$. For example, a 100 binary target masks corresponding with the 100 frames of 1280x720 gray scale images would be stored as 1-bit $100 \times 720 \times 1280 \times 1$. Another sub-image dataset could hold a per-pixel range map containing X, Y, Z coordinate layers. This could be stored in another N-bit sub-image dataset in a $100 \times 720 \times 1280 \times 3$ format. The three channels refer to X, Y, Z coordinates. Number of bits and the data type (integer, float, uint, etc.) is determined by the user. Please refer to **Error! Reference source not found.** for a list of supported entries.

[1] Please find at https://www.emva.org/wp-content/uploads/GenICamPixelFormatValues.pdf

Distribution Statement A: Approved for public release. Distribution is unlimited.

***Table 3.2-1:** Supplemental Image Dataset Attributes*

HDF Key	HDF Value Format	Description
frame_height	integer	Height of the frame in px
frame_width	integer	Width of the frame in px
number_of_channels	integer	Number of channels per frame
channel_type	string	Description of channel storage scheme
pixel_format_name	string	Format type of the image pixels (e.g. Mono8. Refer to GenICamPixelFormats document)
number_of_frames	integer	Number of frames
frame_rate	integer	Frame rate of camera in fps
preprocessing	string	Description of pre-processing done to the imagery
type_of_data	string	Type of data (e.g., real or synthetic)
sensor_name	string	Name of the sensor
sensor_specs	string	Additional sensor specs (e.g., DiaphragmBladeCount)
sensor_identifier	string	Unique ID for sensor
sensor_model	string	Model of the sensor
sensor_type	string	Type of sensor (e.g., visible, MWIR, LWIR)
sensor_subtype	string	Subtype of sensor (e.g., uncooled, cooled)
sensor_location_latitude	float	Sensor latitude at collection start in deg
sensor_location_longitude	float	Sensor longitude at collection start in deg
integration_time	array of float	Integration time of sensor in ms
gain_level	string	Qualitative measure of gain (e.g., high or low)
gain	array of float	Quantitative measure of gain
sensor_known_latency	integer	Sensor latency in ms
gamma_of_imagery	float	Gamma value for viewing imagery
mtf_collection_datetime_UTC	string	Date/time of MTF collection (YYYY-MM-DDThh:mm:ss.ss+00:00)
mtf_of_sensor	array of float	2D array of values describing the radially symmetric MTF of the sensor
psf_of_sensor	array of float	2D array of values describing the radially symmetric PSF of the sensor
sensor_pan_rate	array of float	Sensor pan rate in degrees per second (deg/s)
NUC_datetime_UTC	string	Time of last sensor NUC (YYYY-MM-DDThh:mm:ss.ss+00:00)
NUC_type	string	Type of NUC (e.g., 1, 2, multi-point)
focusing_type_of_camera	string	Type of camera focusing (e.g., manual or auto)
sensor_focal_lengths	array float	Sensor calibrated/effective focal length (mm)
sensor_f_numbers	array float	F# of sensor
sensor_avg_h_fov	float	The average horizontal field-of-view of the sensor (deg)

Distribution Statement A: Approved for public release. Distribution is unlimited.

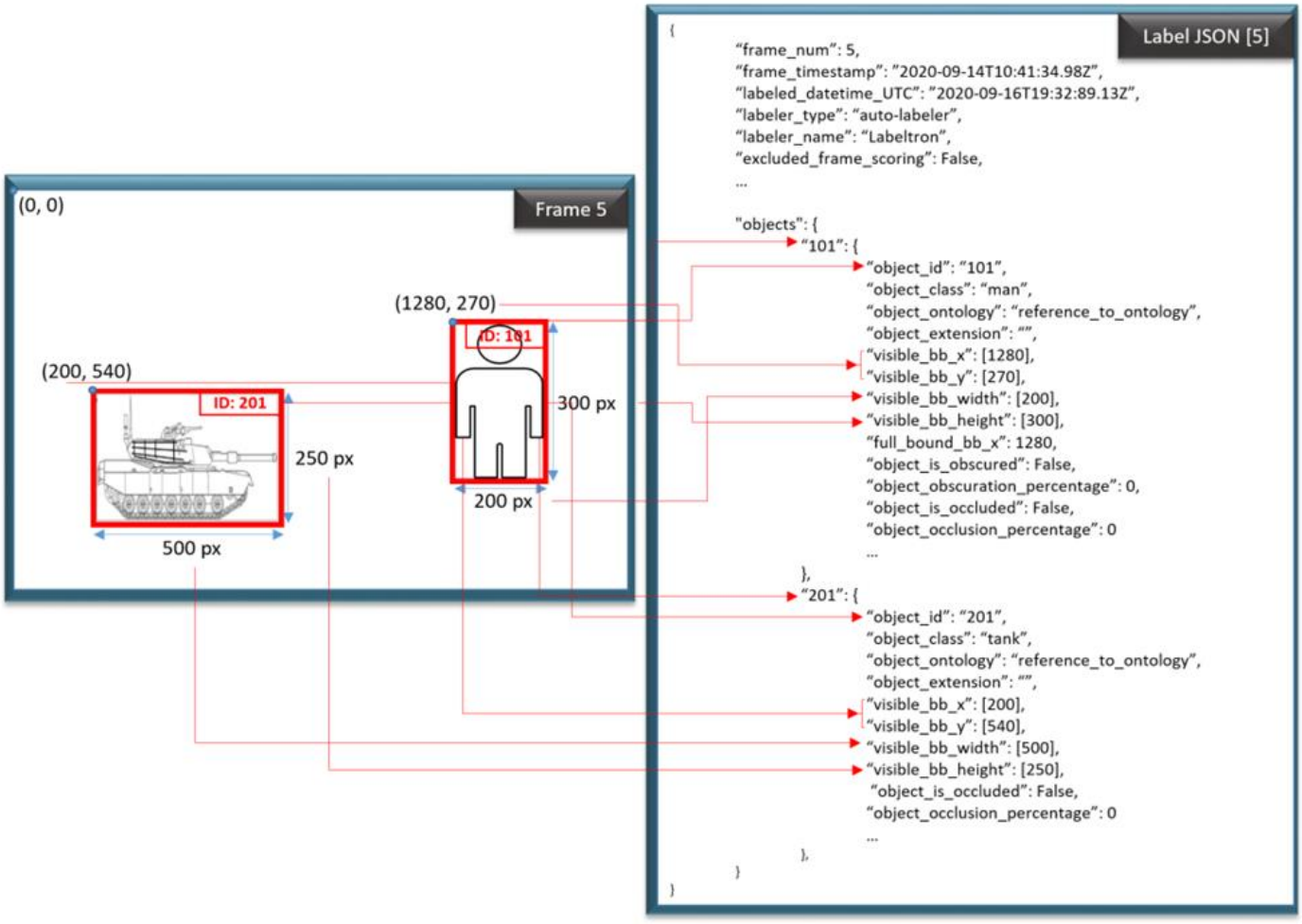
HDF Key	HDF Value Format	Description
sensor_avg_v_fov	float	The average vertical field-of-view of the sensor (deg)
intensity_to_temp_map	array of float	Array describing mapping of pixel intensity to temperature in (K)
well_capacity	float	Well capacity in electrons (e-)
dark_current_density	array of float	Dark current density in nanoAmps/cm^2
total_downstream_noise	float	Total downstream noise in RMS e-
sensor_read_noise	float	Total sensor read noise in RMS e-
sensor_pixel_pitch	float	Sensor pixel pitch value (μm)
sensor_ADC_offset	float	Sensor analog to digital converter offset (e-)
sensor_ADC_bits	float	Sensor analog to digital converter bits
photodetector_fill_ratio	float	Ratio of a pixel's light sensitive area to its total area (e.g., 0.9)
photodetector_width	float	Width of the imaging sensor (mm)
photodetector_height	float	Height of the imaging sensor (mm)
ND_filter_stops	float	Neutral density filter stops
sensor_aperture_size	float	Sensor aperture size (mm)
spectral_quantum_efficiency	array of float	2xN array describing spectral QE where first row is wavelength in μm and second row is dimensionless efficiency value between 0 and 1
horizontal_detector_dimension	float	Horizontal detector dimension in μm
vertical_detector_dimension	float	Vertical detector dimension in μm
horizontal_detector_count	integer	Horizontal detector count
vertical_detector_count	integer	Vertical detector count
noise_equivalent _temperature_difference	float	Noise equivalent temperature difference in K
cut_on_wavelength	float	Cut-on wavelength in μm
color_correction_matrix	array of floats	The matrix used for color correction in the main image dataset
color_correction_offset	array of floats	Offset values used for color correction in the main image dataset
preprocessing_guidelines	string	Description of pre-processing guidelines applied to the entire image dataset (e.g., "used camera sim to add sensor noise")
data_generation_guidelines	string	Guidelines on data creation (e.g., target location radius)
clutter_list	array of string	Array of all clutters that appear in at least one frame of the scene
sensor_relative_coordinates	float	Sensor relative coordinates (x, y, z) to the reference point (m)
sensor_relative_rotation	array of float	Sensor relative rotation (pitch, yaw, roll) to the reference coordinate (deg)
spectral_quantum_efficiency_mean	float	Sensor QE mean
spectral_quantum_efficiency_stdev	float	Sensor QE standard deviation

Distribution Statement A: Approved for public release. Distribution is unlimited.

3.3 Label Dataset

The label dataset stores the per frame information about when imagery was labeled, who labeled it, what was labeled, and who provided quality control for the frame along with per object label information. The label dataset is structured as an array of strings where each string is a serialized JSON describing the label information of the frame at the given index. Within the JSON, there is a list of objects that can be as long as the number of labeled objects shown in each frame. It is important to note that the object ID is supposed to be consistent across all instances of the objects in all frames. This is done to ensure the same object is referenced across frames and also in the synchronous dataset. **Error! Reference source not found.**3.3-1 illustrates an example of how the label dataset JSONs store the label data.

Figure 3.3-1: Label Dataset



Figures are for illustration purposes only, please refer to the CDF text for the full/accurate content.

Distribution Statement A: Approved for public release. Distribution is unlimited.

All of the per object label information used to feed into an AiTR algorithm is stored in the shown JSON format:

Label Dataset JSON Format

Fields highlighted in green are mandatory for CDF compatibility

```
{
    “frame_num”: integer // Frame number corresponding to index of datasets
    “frame_timestamp”: str // Frame timestamp in UTC (YYYY-MM-DDThh:mm:ss.ss+00.00)
    “frame_priority”: integer // Priority of using the frame for labeling and scoring processes. Assigned a numerical from 1 (most priority) to 5
                                (least priority)
    "frame_is_quality_controlled": bool // True if the frame is quality controlled
    “excluded_frame_labeling”: bool // Is the frame intentionally excluded from labeling?
    “excluded_frame_scoring”: bool // Is the frame intentionally excluded from algorithm scoring?
    "excluded_targets": array of string // List of excluded targets in the frame
    "excluded_areas": array of integers  // Array holding user-defined pixel coordinates associated with excluded areas in the frame
    "background": array of strings // Array of strings identifying the background properties
    "objects": {
        <OBJECT_ID>: {
                “object_id”: str // Unique ID to discriminate between objects in a scene
                “object_class”: str // String identifier of the object class
                “object_ontology”: str // Ontology information for the object
                “object_extension”: array of str // Placeholder for recording sub-object description
                “visible_bb_x”: array of integers // Defines the array of x-positions of the top left corners of the object's visible bounding
                                                    boxes from the top left corner of the video frame (px)
                “visible_bb_y”: array of integers // Defines the y-positions of the top left corners of the object's visible bounding boxes from
                                                    the top left corner of the video
                                                    frame (px)
                “visible_bb_width”: array of integers // Width of the object's visible bounding boxes (px)
                “visible_bb_height”: array of integers // Height of the object's visible bounding boxes (px)
                "visible_polygon_bound": array of integers // Array of x-y coordinates specifying the object's visible polygon bounding box
                                                             (px)
                “full_bound_bb_x”: integer // Defines the x-position of the top left corner of object's full bounding box, containing all visible
                                             and occluded portions of the object within the frame boundaries, from left edge of frame (px)
                “full_bound_bb_y”: integer // Defines the y-position of the top left corner of object's full bounding box from top edge of
                                             frame (px)
                “full_bound_bb_width”: integer // Width of object's full bounding box (px)
                “full_bound_bb_height”: integer // Height of object's full bounding box (px)
                "full_polygon_bound": array of integers // Array of x-y coordinates specifying the object's full polygon bounding box,
                                                          containing all visible and occluded portions of the object within the frame
                                                          boundaries (px)
                "pixels_on_target": integer // The number of pixels the target occupies in the frame
                "object_centroid_x": integer // Center of the object x-position from top-left edge of frame
                "object_centroid_y": integer // Center of the object y-position from top-left edge of frame
                “object_is_obscured”: bool // True if the object is obscured, false otherwise
                "object_obscuration_level": string // The qualitative object obscuration (e.g. low, medium, high)
```


Distribution Statement A: Approved for public release. Distribution is unlimited.

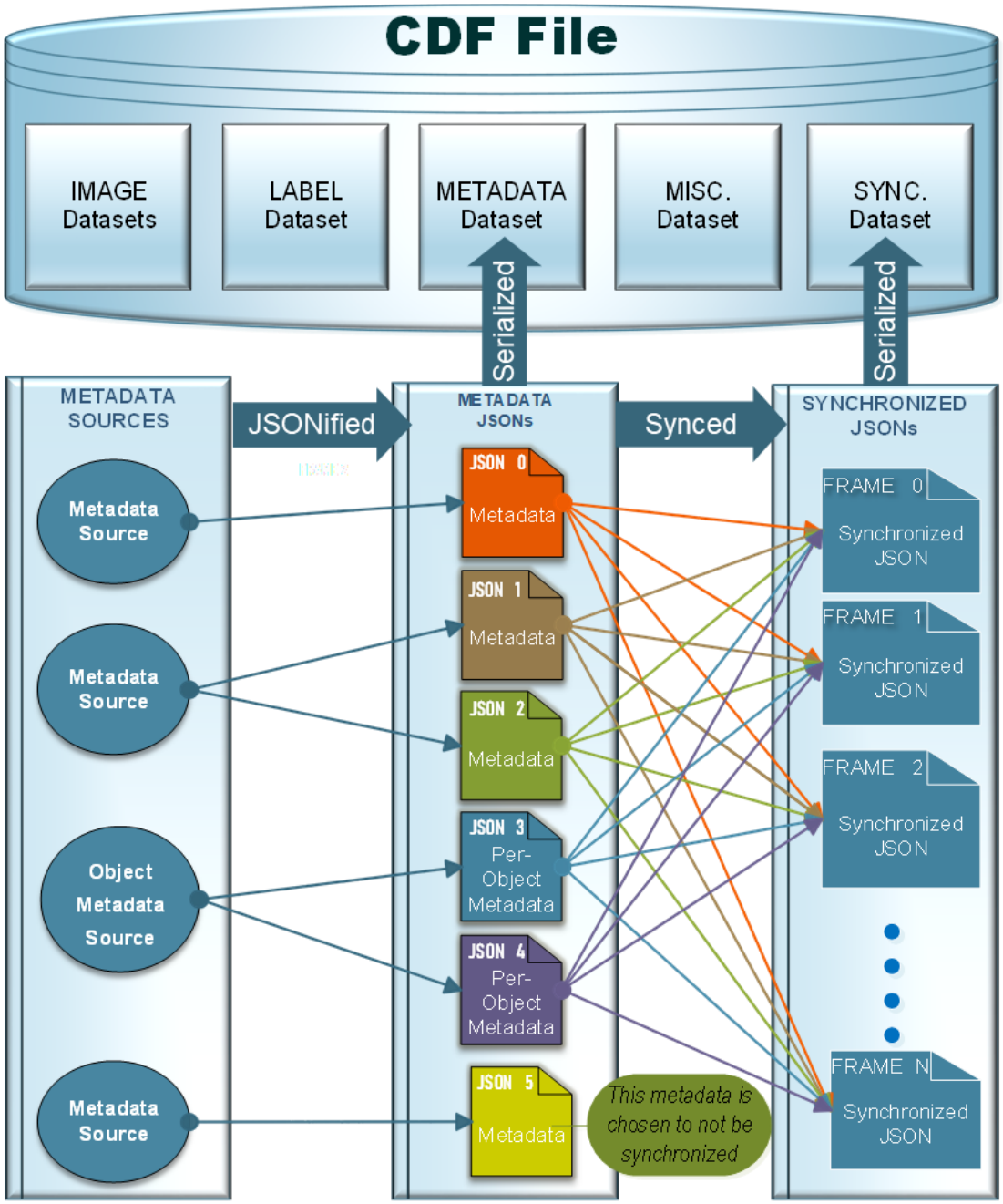
```
                "object_obscuration_percentage": float // Percentage of obscuration of the object (%)
                "object_occlusion_level": string // The qualitative object occlusion (e.g. low, medium, high)
                "object_is_occluded": bool // True if the object is occluded, false otherwise
                "object_occlusion_percentage": float // Percentage of occlusion of the object (%)
                "object_occlusion_foreground": str // The name or object_id of the foreground occluding object
                "is_high_fidelity": bool // Is the object labeled at high fidelity or not (based on the
                                                labeling guidelines)
                "object_relative_rotation": array of float // Object relative rotation (pitch, yaw, roll) to the reference coordinate (deg)
                "edge_of_frame":bool // True if the object is cut-off by the edge of frame
                "inside_bb_average_pixel_value": array of floats // Average pixel value inside an object's visible
                                                                        bounding box
                "outside_bb_average_pixel_value": array of floats // Average pixel value surrounding an object's visible bounding box with
                                                                        the same area
                "inside_bb_stdev_pixel_value": array of floats // Standard deviation pixel value inside an object's visible bounding box
                "outside_bb_stdev_pixel_value": array of floats // Standard deviation pixel value surrounding an object's visible bounding
                                                                        box with the same area
                "labeled_datetime_UTC": str // Date and time of labeling (YYYY-MM-DDThh:mm:ss.ss+00.00)
                "labeler_type": str // Type of labeling method used (e.g. human, machine, metadata, model,
                                        tracker, etc.)
                "labeler_name": str // Name of labeler or labeling algorithm used
                "quality_controlled_datetime_UTC": str // Date and time of quality control
                                                                (YYYY-MM-DDThh:mm:ss.ss+00.00)
                "quality_controller_name": str // Name of quality controller or quality controlling algorithm used
        },
        <OBJECT_ID>: {
                OBJECT 2 LABEL KEYS & VALUES (same format as first object above)
        },
        …
        },
        <OBJECT_ID>: {
                OBJECT N LABEL KEYS & VALUES (same format as first object above)
        }
    }
}
```

3.4 Metadata Dataset

The metadata dataset stores asynchronous data that is not natively synchronized with the frames. This means there is no measurement correlation to every frame in the scene. The metadata dataset is structured as an array of strings. Each string is a serialized JSON describing a single asynchronous metadata source. Figure 3.4-1 illustrates how metadata is stored within the serialized JSONs.

Distribution Statement A: Approved for public release. Distribution is unlimited.

Figure 3.4-1: Metadata Storage Process



It is important to note that the measurements for each data type (“data_name”), and thus JSON, is stored as an array of values under the “measurements” key. The measurements array combined with the array of synchronizing variable values, “sync_variable_array” and the interpolation algorithm, “interpolation_algorithm”, can be used to interpolate the metadata and populate the synchronous dataset JSONs. This will be covered more in the next section.

Distribution Statement A: Approved for public release. Distribution is unlimited.

The JSON format below specifies how to store metadata that does not apply to the individual objects in each frame (a/k/a asynchronous metadata).

General metadata JSON Format

Fields highlighted in green are mandatory for CDF compatibility

```
{
    "data_name": str, // The key name of the metadata which is used when storing the interpolated metadata values in the JSONs of the synchronous dataset
    "data_source": str, // Name of sensor or source of the metadata
    "data_references": str. // A placeholder for storing references and/or URLs
    "measurement_units": str, // String describing the units used in measurements
    "sync_variable_type": str, // The type of the variable used to synchronize data with frames.
    "has_been_synchronized": bool, // Describes if the data has been synchronized and copied to the synchronous dataset through interpolation
    "interpolation_algorithm": str, // The type of interpolation algorithm used
    "sync_variable_array": array of values // The corresponding timestamps/frames/etc. used to synchronize the metadata with the frames
    "measurements": array of values // The measurements of the data type which corresponds to the synchronizing variable array
    "measurement_known_latency": integer, // Latency of the data source measurements in milliseconds (ms)
    "measurement_uncertainty": str, // Description and measurement of uncertainty
    "measurement_standard_deviation": float // The standard deviation of the measurements from the data source
}
```

The JSON format below specifies how to store metadata that applies to the individual objects in each frame. The format is the same as the header metadata JSON format except that an object class and ID must be specified.

Note: after synchronization, this metadata will move to the per frame section of the synchronous dataset denoted as <OBJECT_ID>. Refer to <PER-FRAME_DATA_NAME> *for more information.*

General object metadata JSON Format:

```
{
    "object_id": str, // Unique ID to discriminate between objects in a scene
    "data_name": str, // The key name of the metadata which is used when storing the interpolated metadata values in the JSONs of the synchronous dataset
    "data_source": str, // Name of sensor or source of the metadata
    "data_references": str. // A placeholder for storing references and/or URLs
    "measurement_units": str, // String describing the units used in measurements
    "sync_variable_type": str, // The type of the variable used to synchronize data with frames.
    "has_been_synchronized": bool, // Describes if the data has been synchronized and copied to the synchronous dataset through interpolation
    "interpolation_algorithm": str, // The type of interpolation algorithm used
    "sync_variable_array": array of values // The corresponding timestamps/frames/etc. used to synchronize the metadata with
```


Distribution Statement A: Approved for public release. Distribution is unlimited.

```
                                                    the frames
    "measurements": array of values // The measurements of the data type which corresponds to the synchronizing variable
                                                    array
    "measurement_known_latency": integer, // Latency of the data source measurements in
                                                    milliseconds (ms)
    "measurement_uncertainty": str, // Description and measurement of uncertainty
    "measurement_standard_deviation": float // The standard deviation of the measurements from the data source
}
```

Permitted Metadata Keys/Values:

The following tables show the standardized metadata names and corresponding value format along with a short description. It should be noted that the permitted metadata "data_name" column specifies the string values that should be stored under the key "data_name". **Error! Reference source not found.**.4-1 corresponds to metadata that should be stored in the general per frame metadata JSON format. **Error! Reference source not found.** corresponds to the metadata values that should be stored in the general per object metadata JSON format.

Table 3.4-1: Permitted General "data_name" Values

Permitted Metadata "data_name"	Value Format	Description
sensor_latitude	float	Sensor latitude in deg
sensor_longitude	float	Sensor longitude in deg
sensor_elevation	float	Sensor elevation in m
sensor_roll	float	Sensor roll in deg
sensor_pitch	float	Sensor pitch in deg
relative_sensor_pitch	float	Relative sensor pitch to the gimbal in deg
sensor_yaw	float	Sensor yaw in deg
relative_sensor_yaw	float	Relative sensor yaw to the gimbal in deg
sensor_focal_length	float	Sensor calibrated/effective focal length (mm)
sensor_f_number	float	F# of sensor
sensor_h_fov	float	Horizontal sensor field-of-view (FOV) in deg
sensor_v_fov	float	Vertical sensor fFOV in deg
lidar_measurement	array of float	Lidar measurements of scene in m
cloud_cover_percentage	float	Percentage of cloud coverage (%)
cloud_base_height	float	Height of the cloud base above mean sea level (m)
ambient_temperature	float	Temperature outside in K
dew_point_temperature	float	Dew point temperature in K

Distribution Statement A: Approved for public release. Distribution is unlimited.

relative_humidity	float	Relative humidity (%)
Permitted Metadata "data_name"	**Value Format**	**Description**
shortwave_upwelling	float	Atmospheric upwelling (W/m^2)
longwave_upwelling	float	Atmospheric upwelling (W/m^2)
shortwave_downwelling	float	Atmospheric downwelling (W/m^2)
longwave_downwelling	float	Atmospheric downwelling (W/m^2)
instantaneous_turbulence_condition	float	Instantaneous turbulence, Cn^2, condition (m^-2/3)
instantaneous_wind_speed	float	Instantaneous wind speed (m/s)
instantaneous_wind_direction	float	Instantaneous wind direction (deg)
solar_azimuth	float	Solar azimuth in deg
diffuse_solar_irradiance	float	Diffuse solar irradiance (W/m^2)
global_hemispheric_solar_irradiance	float	Global hemispheric solar irradiance (W/m^2)
direct_solar_irradiance	float	Direct solar irradiance (W/m^2)
solar_zenith	float	Solar zenith in deg
precipitation_type	string	Precipitation type rain, snow, etc.
precipitation_rate	float	Rate of precipitation type rain, snow, etc. (mm/hour)

Table 3.4-2: Permitted Per-object Metadata

Permitted Per-object Metadata "data_name"	Value Format	Description
object_latitude	float	Object latitude in deg
object_longitude	float	Object longitude in deg
object_elevation	float	Object elevation in meters (deg)
object_aspect	float	Degrees of rotation (Front = 0, Vehicle POV 0-359 deg)
object_speed	float	Absolute velocity of the target relative to the ground (m/s)
object_velocity_east	float	Velocity of the target in the eastward direction (m/s)
object_velocity_north	float	Velocity of the target in the north direction (m/s)
object_velocity_up	float	Velocity of the target in the upward direction (m/s)
object_heading	float	Object heading direction (deg)
object_relative_azimuth	float	Azimuth of the object relative to the sensor in deg
object_relative_elevation	float	Elevation of the object relative to the sensor in m
object_relative_range	float	Range of object relative to the sensor in m
object_slant_range	float	The distance along the relative direction between the camera and the target (mm)
object_relative_coordinates	float	Object relative coordinates (x, y, z) to the reference point

Distribution Statement A: Approved for public release. Distribution is unlimited.

		(m)
Permitted Per-object Metadata "data_name"	**Value Format**	**Description**
object_relative_rotation	Array of float	Object relative rotation (pitch, yaw, roll) to the reference coordinate (deg)
object_dimensions	array of float	Vector of dimensions of the given object in m
object_solar_loading	string	Qualitative or quantitative description of the solar loading of the object
object_minimum_temperature	float	Object minimum temperature (K)
object_maximum_temperature	float	Object maximum temperature (K)
object_temperature_bias	float	Object temperature bias (K)
object_temperature_gain	float	Object temperature gain (K)
object_is_muddy	bool	True if the object is muddy
object_is_dusty	bool	True if the object is dusty
object_is_wet	bool	True if the object is wet
object_ground_reflection	string	Qualitative or quantitative description of the ground reflection on the object
object_sky_reflection	string	Qualitative or quantitative description of the sky reflection on the object
object_total_irradiance	float	Total irradiance on the object (W/cm^2)
object_is_exercised	bool	The engine is running or appears to be running
object_engine_status	string	Describes engine status (i.e. idling, accelerating, etc.)

3.5 Synchronous Dataset

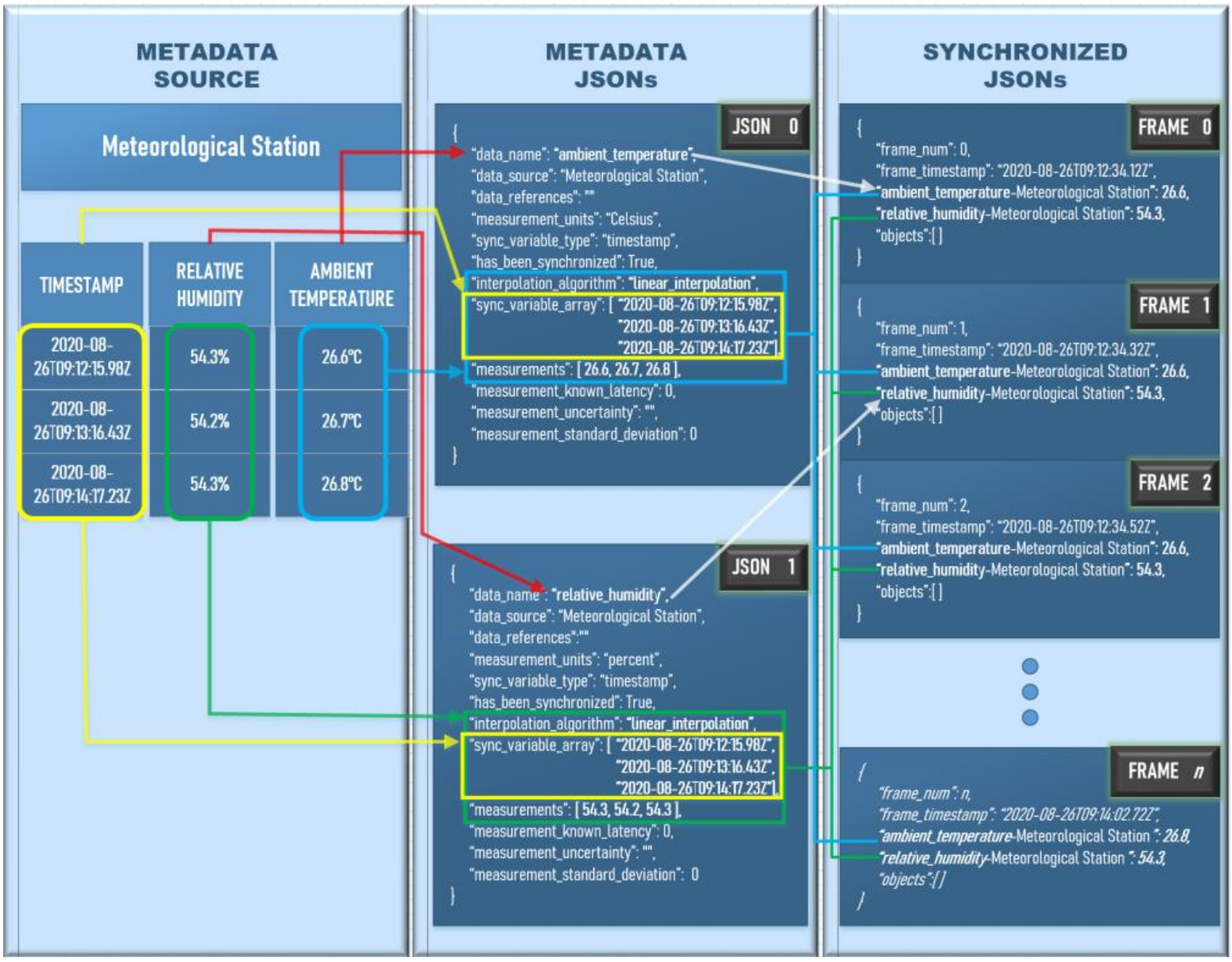
It is important to note that in the metadata dataset, the measurements for each data type ("data_name"), and thus JSON, is stored as an array of values under the "measurements" key. The measurements array combined with the array of synchronizing variable values, "sync_variable_array", and the interpolation algorithm, "interpolation_algorithm," can be used to interpolate the metadata and populate the synchronous dataset JSONs.

The synchronous (sync) dataset stores all synchronized metadata for each frame which includes both per frame metadata and per object metadata. Similar to the label dataset, all information is stored in a JSON format for every frame in the scene. As shown in **Error! Reference source not found.**, for each frame in the scene, there is a corresponding label JSON and synchronous data JSON. The first column is the metadata source. The middle column describes the JSONs that would be in the metadata dataset (asynchronized). The last column shows how this data is mapped into synchronized JSONs. These synchronized JSONs are serialized and put into an array of strings such that each frame has one

Distribution Statement A: Approved for public release. Distribution is unlimited.

matching JSON of metadata. Please note that in the synchronized JSONs the “data_source” is appended to the “data_name” resulting in “data_name-data_source” key format. **Error! Reference source not found.** illustrates the similar per object synchronization process.

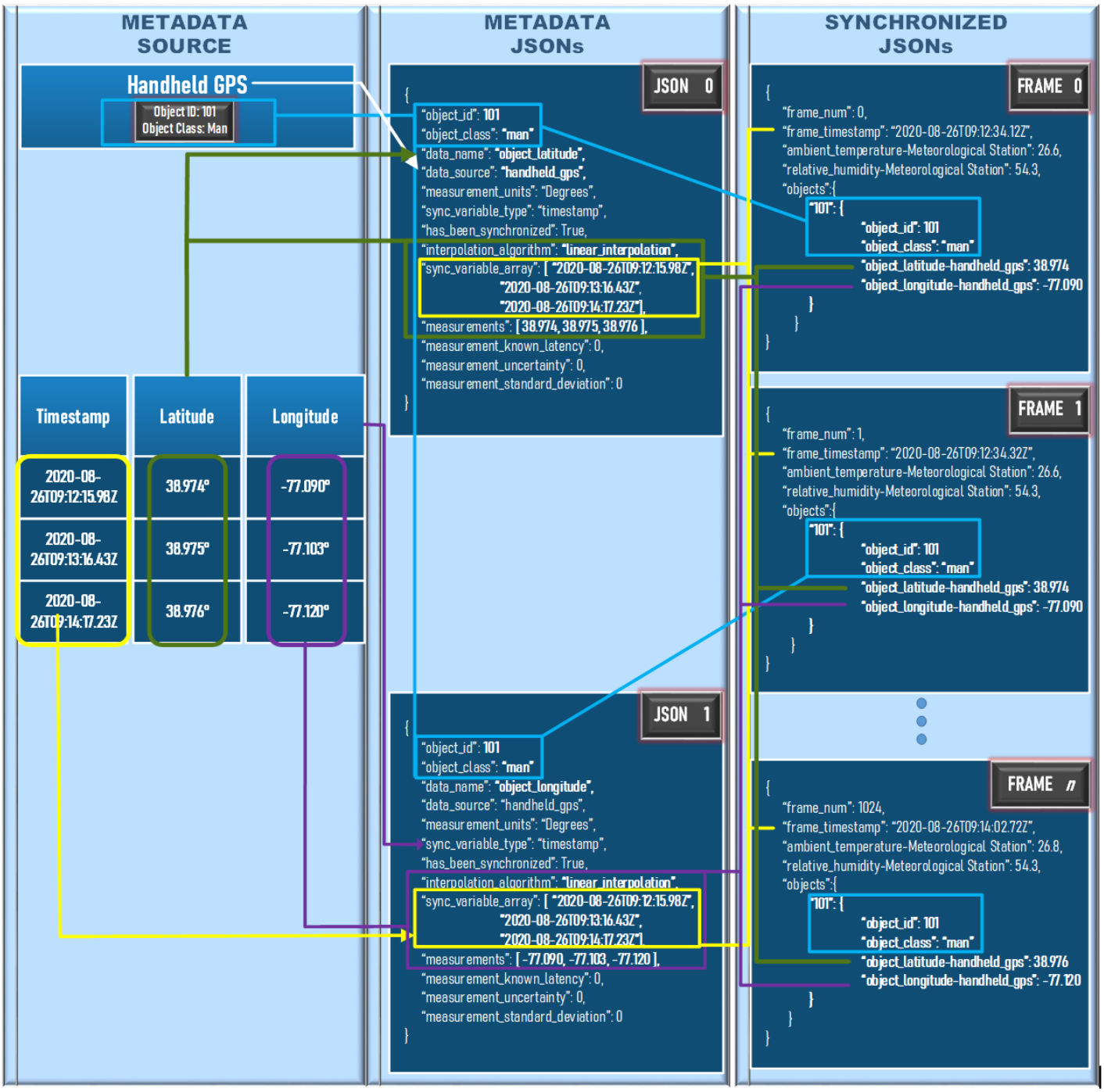
Figure 3.5-1: Metadata Synchronization Process



Figures are for illustration purposes only, please refer to the CDF text for the full/accurate content.

Distribution Statement A: Approved for public release. Distribution is unlimited.

Figure 3.5-2: Per Object Synchronization Process



Figures are for illustration purposes only, please refer to the CDF text for the full/accurate content.

Distribution Statement A: Approved for public release. Distribution is unlimited.

The format of JSON along with all of the possible key-value pairs are given below:
Fields highlighted in green are mandatory for CDF compatibility

{

“frame_num”: integer // Frame number corresponding to index of image and label datasets

“frame_timestamp”: str // Frame timestamp in UTC (YYYY-MM-DDThh:mm:ss.ss+00:00)

<data_name-data_source>: <INTERPOLATED_VALUE> // Placeholder for metadata that has been synchronized from the metadata dataset

<data_name-data_source>: <INTERPOLATED_VALUE> // Placeholder for metadata that has been synchronized from the metadata dataset

…

<data_name-data_source>: <INTERPOLATED_VALUE> // Placeholder for metadata that has been synchronized from the metadata dataset

"objects": {

<OBJECT_ID>: {

“object_id”: integer // Unique ID to discriminate between objects in a scene

<data_name-data_source>: <INTERPOLATED_VALUE> // Placeholder for metadata that has been synchronized from the metadata dataset

<data_name-data_source>: <INTERPOLATED_VALUE> // Placeholder for metadata that has been synchronized from the metadata dataset

…

<data_name-data_source>: <INTERPOLATED_VALUE> // Placeholder for metadata that has been synchronized from the metadata dataset

},

<OBJECT_ID>: { *OBJECT 2 SYNCED METADATA KEYS & VALUES* },

<OBJECT_ID>: { *OBJECT 3 SYNCED METADATA KEYS & VALUES* },

…

<OBJECT_ID>: { *OBJECT N SYNCED METADATA KEYS & VALUES* }

}

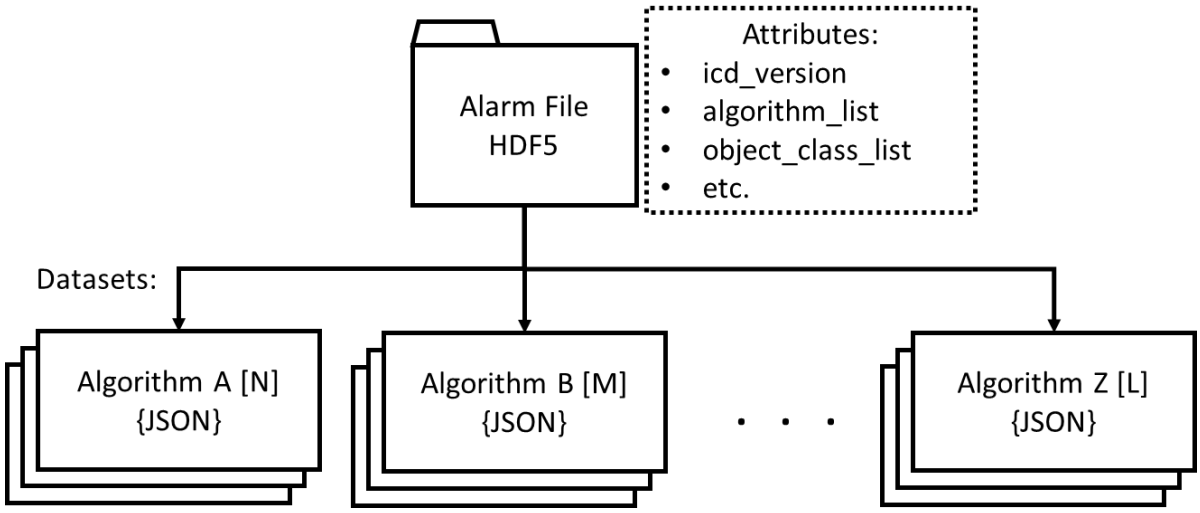
}

Distribution Statement A: Approved for public release. Distribution is unlimited.

4. Alarm Dataset

This dataset stores information about algorithm performance in identifying objects of interest in the scene. The alarm data is stored in its own HDF file separate from the rest of the data. As illustrated in **Error! Reference source not found.**, the alarm file could contain numbers of datasets corresponding to every instance of an algorithm run. Each algorithm dataset is composed of layers corresponding to a JSON which contains alarms for each frame of the image data that was subject to the algorithm. Alarm file group attributes contain information mostly applicable to the entire file such as algorithm_list, test_file_hash, data_source, etc.

Figure 4-1: Alarm File Architecture



4.1 Alarm HDF5 Group Attributes

Group attributes are used to specify higher level information that are associated with the algorithms and the dataset. **Error! Reference source not found.** provides a full list of these attributes. **Error! Reference source not found.** provides an alarm JSON example.

Table 4.1-1: **Alarm Dataset Group Attributes**
Fields highlighted in green are mandatory for CDF compatibility

HDF5 Key	HDF5 Value Format	Description
data_source	string	String describing the location of the data source (e.g. URL, path)
test_file_hash	array of string	Hash for HDF5 test files in database
schema_guide_version	string	Schema guide version used for creating the file

Distribution Statement A: Approved for public release. Distribution is unlimited.

Distribution Statement A: Approved for public release. Distribution is unlimited.

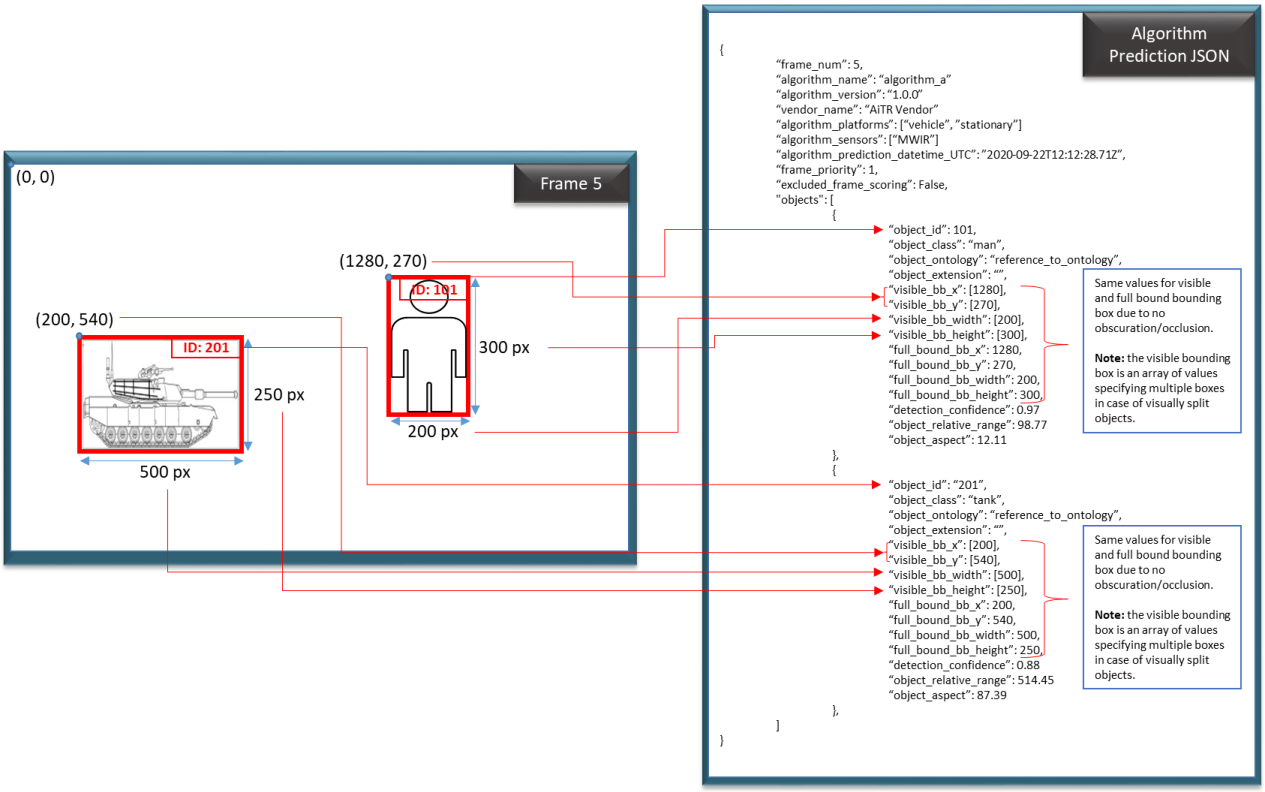
Distribution Statement A: Approved for public release. Distribution is unlimited.

HDF5 Key	HDF5 Value Format	Description
ontology	string	Ontology name, version, base IRI used for assigning object classes
hdf5_creation_datetime_UTC	string	Creation time of HDF5 in UTC (YYYY-MM-DDThh:mm:ss.ss+00.00)
hdf5_creation_datetime_local	string	Creation time of HDF5 in local time (YYYY-MM-DDThh:mm:ss.ss+hh:mm)
hdf5_modified_datetime_UTC	string	Modification time of HDF5 in UTC (YYYY-MM-DDThh:mm:ss.ss+00.00)
hdf5_modified_datetime_local	string	Modification time of HDF5 in local time (YYYY-MM-DDThh:mm:ss.ss+hh:mm)
hdf5_phase	integer	Current phase information of the HDF5 file
algorithm_list	array of string	List of algorithm_name/s contained within the alarm file
organization_name	string	Name of the organization
division_name	string	Division or lab name
branch_name	string	Name of branch
project_type	string	Global ID intended to uniquely identify all files associated with a project
project_name	string	Name of project
collection_name	string	Unique name for the collection (e.g. a combination of project name, collection site, collection time period)
type_of_data	string	Type of data (e.g., real or synthetic)
site_identifier	string	Unique ID of collection site
collection_datetime_UTC	string	Start date and time of collection in UTC (YYYY-MM-DDThh:mm:ss.ss+00.00)
collection_datetime_local	string	Start date and time of collection in local time (YYYY-MM-DDThh:mm:ss.ss+hh:mm)
scene_number	integer	Number of scene from the collection plan
scene_look_number	integer	Number of look in case of repeat collections of the scene
dataset_poc	string	Contact information for POC familiar with the dataset/collection
object_class_list	array of string	Array of all object classes that appear in at least one frame of the scene

Distribution Statement A: Approved for public release. Distribution is unlimited.

4.2 Predictions JSON Format

Figure 4.2-1: Alarm JSON Example



Figures are for illustration purposes only, please refer to the CDF text for the full/accurate content.

Predictions JSON Format

Fields highlighted in green are mandatory for CDF compatibility

{

"frame_num": integer // Frame number corresponding to index of datasets

"algorithm_name": str // Target recognition algorithm used in predictions

"algorithm_version": str // Version of the algorithm

"vendor_name": str // Vendor name of algorithm model creator/source

"algorithm_platforms": array of str // List of platforms algorithm was created for

"algorithm_sensors": array of str // List of sensors the algorithm was created for

Distribution Statement A: Approved for public release. Distribution is unlimited.

“algorithm_prediction_datetime_UTC”: str // Date and time of prediction (YYYY-MM-DDThh:mm:ss.ss+00.00)
"model_AP": array of float // Average precision
"model_recall": array of float // Average Recall
"model_AUC": array of float // Area under the curve for ROC
“excluded_frame_scoring”: bool // Is the frame intentionally excluded from algorithm scoring?
“frame_priority”: integer // Priority of using the frame for labeling and scoring processes. Assigned a numerical value from 1 (most priority) to 5 (least priority)
"objects": [
{
“object_id”: integer // Unique ID to discriminate between repeats of an object class in a scene
“object_class”: str // String identifier of the object class
“object_ontology”: str // Ontology information for the object
“object_extension”: array of str // Placeholder for recording list of sub-object description
“predicted_bb_x”: integer // Defines the x-position of the top left corner of the object's predicted bounding box from the top left corner of the video frame (px)
“predicted_bb_y”: integer // Defines the y-position of the top left corner of the object's predicted bounding box from the top left corner of the video frame (px)
“predicted_bb_width”: integer // Width of the object's predicted bounding box (px)
“predicted_bb_height”: integer // Height of the object's predicted bounding box (px)
“object_confidence_score”: float // The overall confidence score for the object on the range (0,1.0) (e.g. the confidence of the last stage in the process of detector, classifier, tracker, etc.)
“object_confidence_task”: str // The algorithm’s task for the given object confidence score (e.g. “detector”, “classifier”, “tracker”)
“object_confidence_metric”: str // Metric of the algorithm confidence score for the object (e.g. ”mean average precision”, “accuracy”, “F1 score”)
“object_predicted_class”: str // The predicated object class by the algorithm
“object_detection_confidence”: float // The algorithm detection confidence score for the object on the range (0,1.0)
“object_classification_confidence”: float // The algorithm classification confidence score for the object on the range (0,1.0)
“object_tracker_confidence”: float // The algorithm tracking confidence score for the object on the range (0,1.0)
"predicted_polygon_bound": array of integers // Array of x-y coordinates specifying the object's predicted polygon bounding box (px)
“object_relative_range”: float // Range of object relative to the sensor in meters (m)
“object_aspect”: float // Degrees of Rotation (Front = 0, Vehicle POV 0-359 deg)
"object_latitude": float // Object latitude in degrees (deg)
"object_longitude": float // Object longitude in degrees (deg)
"object_elevation": float // Object elevation in meters (deg)
"object_aspect": float // Degrees of Rotation (Front = 0, Vehicle POV 0-359 deg)

Distribution Statement A: Approved for public release. Distribution is unlimited.

"object_velocity": float // Absolute velocity of the target relative to the ground (m/s)
"object_relative_azimuth": float // Azimuth of the object relative to the sensor in degrees (deg)
"object_relative_elevation": float // Elevation of the object relative to the sensor in meters (m)
},
{ 2nd *OBJECT PREDICTION KEYS & VALUES (same format as first object above)* },
{ 3rd *OBJECT PREDICTION KEYS & VALUES (same format as first object above)* },
{ Nth *OBJECT PREDICTION KEYS & VALUES* }
]
}

Distribution Statement A: Approved for public release. Distribution is unlimited.

A. Acronyms

Acronyms

Acronym	Definition
AiTR	Aided Target Recognition
AI2C	Artificial Intelligence Integration Center
API	Application Programming Interface
ARL	Army Research Laboratory
ATR	Automatic Target Recognition
ATRWG	ATR Working Group
CDF	Common Data Format
deg	Degrees
deg/s	Degrees per Second
DoD	Department of Defense
DUSD (S&T/SS)	Deputy Under Secretary of Defense for Science and Technology Sensor Systems
e-	Electrons
FOV	Field-of-View
Frames per second	FPS
GPS	Global Positioning System
GPU	Graphics Processing Unit
HDF5	Hierarchical Data Format 5
ID	Identification
IR	Infrared
IRI	Internationalized Resource Identifier
JSON	JavaScript Object Notations
K	Kelvin
LWIR	Longwave Infrared
m	Meter
m/s	Meters per Second
mm	Millimeter
ms	Milliseconds
MTF	Modulation Transfer Function
MWIR	Midwave Infrared
NUC	Non-Uniformity Correction

Distribution Statement A: Approved for public release. Distribution is unlimited.

POC	Point of Contact
PSF	Point Spread Function
px	Pixels
QE	Quantum Efficiency
RMS	Root-Mean-Squared
sRGB	Standard Red, Green, Blue
Sync	Synchronous
µm	Micrometer

Distribution Statement A: Approved for public release. Distribution is unlimited.

Revision History

Track revision history in the table below. Rev. 1.00 is the Initial Release including the version of the template used to create the Initial Release. Later revisions summarize changes made. These instructions can be hidden if they need to be retained.

- *Minor revisions are typos or grammatical errors that do not change the intent of the signed document. Minor revisions are recorded with a 1/100th increase of the revision number (X.00 to X.01).*
- *Major revisions are changes that impact the scope, cost, schedule, deliverables, or intent of the document. Major revisions are recorded with an integer change to the revision number (1.XX to 2.00).*

1.00	MM-DD-YYYY		Initial Release, based on template Revision {x.x}

Distribution Statement A: Approved for public release. Distribution is unlimited.