Export 3d Annotations

## Motivation

Sadly there’s no standardized format yet for annotations in the autonomous driving industry, so for our annotation projects the customer expects the annotated data to be delivered in a format specified by them. Therefore we have to write project specific software to convert the annotations.

## Your Task

You Task is to write a converter according to the attached fictional specifications. You are free to use whatever technology, frameworks or languages that help you solve the task. The raw annotations should be loaded and saved from and to a json-file but the code should be modular enough to allow loading annotations from a web service in the future.

Please take some time to explain less obvious design choices, either by email or when you are here for your test day.

You don’t have to produce production ready code but work towards it and note what would be missing to make it ready.

## Potential Pitfalls:

* annotations are always prone to labeling and configuration errors. so please think of basic validations you would add to the script. In this case, the script should log meaningful error messages

## Specifications for the converted annotations

In this project we have to annotate bicycles and pedestrians.

A typical converted file should look like this:

|  |
| --- |
| {  "FRAMES": [{  "FRAME\_ID": ”,  "BICYCLES": [  {  "BICYCLE\_ID": 0,  "POSITION": [0,0,0],  "ORIENTATION": [0,0,0,1],  "SIZE": [1,1,1],  "STATUS": 2,  "RIDER": null,  "TYPE": 1  }  ],  "HUMANS": [  {  "HUMAN\_ID": 1,  "POSITION": [0,0,0],  "ORIENTATION": [0,0,0,1],  "SIZE": [1,1,1],  "WEARS\_HELMET": 0,  "AGE": 1  }  ]  }]  } |

A customer would generally provide a specification as follows:

|  |  |  |
| --- | --- | --- |
| **NAME** | **TYPE** | **DESCRIPTION** |
| FRAMES | ARRAY[FRAME\_OBJECT] | List of frames |
| FRAME\_ID | INTEGER | id of the frame |
| BICYCLES | ARRAY[BICYCLE\_OBJECT] | list of bicycles on the frame |
| HUMANS | ARRAY[HUMAN\_OBJECT] | list of humans on the frame |
| BICYCLE\_ID | INTEGER | Identifier of the bicycle. When the same object appears on multiple frames, it should have the same BICYCLE\_ID. Numbering should be consecutive and bicycles which first appear on earlier frames should have smaller ids |
| POSITION | TUPLE[FLOAT, FLOAT, FLOAT] | Position of the center of the box containing the object (front, left, up in that order) |
| ORIENTATION | TUPLE[FLOAT,FLOAT,FLOAT,FLOAT] | orientation of the box containing the object as a quaternion |
|  | if you are not familiar with the use of quaternions, you can also choose a different way to represent the orientation | |
| SIZE | TUPLE[FLOAT, FLOAT, FLOAT] | width, length and height of the box containing the object |
| RIDER | INTEGER | null | HUMAN\_ID of the rider of the bicycle,  null if there is no rider on the bike |
| STATUS | INTEGER | 0: bicycle is driving  1: bicycle is stopped  2: bicycle is parked (only if the bicycle have no rider |
| TYPE | INTEGER | 0: normal bicycle  1: motorized bicycle |
| HUMAN\_ID | INTEGER | Same as BICYCLE\_ID, but for humans |
| WEARS\_HELMET | INTEGER | null | 0: human wears no bike helmet  1: human wears bike helmet  null: human does not ride a bicycle |
| AGE | INTEGER | 0: adult  1: child |

## Examples of annotations in UAI format

|  |
| --- |
| {  "annotationId" : "4ef8a8d0-8520-4462-9d5b-fb1ac560daac",  "frameId" : "12",  "temporalId" : "981b177f-3ab1-457f-b447-445aef76f6c1"  "label" : "BICYCLE",  "attributes" : {  "status" : "stopped",  "type" : “normal”,  },  "x" : 2,  "y" : 0.3,  "z" : -10  "roll" : 0.3  "pitch" : -0.8  "yaw" : 14  "length" : 2  "height" : 1  "width" : 0.4  } |

|  |
| --- |
| {  "annotationId" : "1757171e-dc05-414c-80ac-9f7d8dcda85e",  "frameId" : "12",  "temporalId" : "44224f47-c502-45c8-a2b3-e092b9a2e3ce"  "label" : "HUMAN",  "attributes" : {  "age" : "adult",  "wears\_helmet" : true,  “rides\_on\_bicycle”: ”981b177f-3ab1-457f-b447-445aef76f6c1”  },  "x" : 1.8,  "y" : 0.3,  "z" : -8.5  "roll" : 0.2  "pitch" : -1  "yaw" : 44  "length" : .5  "height" : 2  "width" : 0.4  } |

We use a right handed, y-up coordinate system for positions, the yaw, pitch, roll convention for rotations and (width, height, length) for dimensions. Please note that the yaw, pitch, roll convention is usually formulated as z-y'-x'' for a z-up system, which corresponds to y-z'-x'' in our system. Angles are saved as degrees.

|  |
| --- |
| **UAI coordinate system**  Y (height, yaw axis)  ^ 0--------1  | / | /|  |-----> X (length, 3-------2 | |  / roll axis) | 4------|--5 facing this way (0 yaw)  v |/ |/ --------------------------->  Z (width, pitch axis) 7--------6 |

The ‘annotationId’ is unique uuid for each annotation. The ‘temporalId’ references a real world object and is consistent over different frames (also a uuid). ‘label’ is the name of the class of the annotation.

The Attributes-dictionary has the following entries:

* for label HUMAN:
  + wears\_helmet boolean value
  + age: string, one of the values “adult” or “child”
  + “rides\_on\_bicycle”: string, temporalId of the BICYCLE the human rides on. empty string if the human doesn’t rides a bicycle
* for label BICYCLE
  + status: string, one of the values "parked", "stopped","moving"
  + type: string, one of the values "normal","motorized"