Proposal for Creating Open-Source Datasets of Embodied AI

Dear colleagues from universities, scientific research institutes and enterprises across the country:

Embodied AI is the current cutting-edge direction of intelligence science and is considered the only channel to achieve general AI. Developed countries led by the United States and many technology giants represented by OpenAI and Nvidia all regard embodied intelligence as an important strategic development and have begun their deployment with huge investment.

In recent years, robot technology has developed rapidly, profoundly affecting our production and daily life. From industrial manufacturing to family life, from deep-sea exploration to space experiments, robots are appearing more and more frequently in various fields. Robots are physical carriers of embodied intelligence. Embodied intelligence further endows robots with brains, senses, and experience capabilities, enabling them to continuously learn and improve through repeated interactions with the environment. The embodied intelligent robot system will subvert the traditional single working mode of dedicated and weakly interactive robots, and is expected to be implemented on a large scale in open and unstructured scenarios to meet the diverse needs of humans.

In the era of large models, by increasing the amount of data and expanding the model scale, model performance can be continuously improved. In order to build a large basic model of embodied AI and build an application platform with large-scale effects for embodied AI, it's urgent to create an open source large-scale and high-quality robot perception and operation data set.

Just as Stanford University's ImageNet promotes computer vision research, as an important part of the country's strategic scientific and technological strength, we hope to rely on the "China Computing Net" and "OpenI" open-source ecology to take the lead in creating an open-source dataset with the same influence in the field of embodied AI: ARIO (All Robots In One).

At present, Google (US) has firstly released the Open X-Embodiment data sets and based on which an Embodied AI control basic model RT-X is trained. It shows good generalization performance across scenarios, multi-tasks, cross-platforms, etc. and generally exceeds previous skill levels based on specific scenarios and datasets.

In order to promote cutting-edge exploration and industrial application in the field of embodied intelligence in China, PengCheng Laboratory, together with AgileX Robotics, Sun Yat-sen University, Southern University of Science and Technology, etc., started it, and supported by a series of scientific research institutes, enterprises, institutions and individuals in the industry, such as the University of Hong Kong, Technical University of Munich, JD Explore Academy, D Robotics, Dataa Robotics, Zhiyuan Robotics, and the Chinese University of Hong Kong, to jointly create the first large-scale, multi-modal embodied intelligence dataset in China: ARIO, that covers multiple scenarios, skills, tasks, and platform type.

Compared with Open X-Embodiment, ARIO will be the world's first dataset to include five modalities (Image, point cloud, text, touch and voice), covering both fields such as service and industry and supporting rich application scenarios. Hence, we call on all parties to:

1. Establish unified collection and open-source standards for robot perception and operation data to ensure data standardization.
2. Strengthen data security and privacy protection to ensure that open-source data does not infringe personal privacy and business secrets.
3. Encourage universities, scientific research institutions and enterprises to actively participate in data collection, sharing and collaborative research, jointly build a high-quality data set ecosystem, and jointly promote the innovative development of embodied intelligence technology.

Let us work together to promote the creation of ARIO, and to contribute knowledge and strength to the prosperity and development of the embodied intelligence and robotics industry!

Sincerely,
28th March, 2024
PengCheng Laboratory, AgileX Robotics, Sun Yat-sen University, Southern University of Science and Technology

Appendix 1. Data format description

1.1 Data Structure

The whole data file structure is divided into: collection——series——episode.
Collection refers to a data set sample submitted and uploaded at one time, which may include different scenarios and robot types.
Series refers to a series of data collected by the same scene and the same robot, such as a series of data collected by a dual-arm robot in the kitchen, which may include different tasks.
Episode refers to a specific action task, such as holding a water glass. In an episode, sensors collect data. Each sensor can collect data by itself according to its own frequency, but it must be based on the same timestamp. The sample file structure is as follows:

```
Collection (A sample of the dataset submitted at one time)
 │ commit.yaml (Information and statement of the author)
 │
 ├─series-1 (The same scene, the same robot)
 │ │calibration_1.yaml (Camera 1 calibration parameters)
 │ │calibration_cam1_lidar1.yaml (Camera 1 and Lidar 1 calibration parameters)
 │ │IMU.pdf (IMU instruction)
 │ │information.yaml (Scene description and robot information)
 │ │Touch.pdf  (Touch sensor instruction)
 │ │AgileX Robot instruction.pdf
 │ │
 │ ├─task-1 (One task. For example: pick up an Apple)
 │ │ │  description.yaml (instruction)
 │ │ │  task_record.mp4 (Video of each task)
 │ │ │
 │ │ ├─episode-1 (A task, such as: getting a water cup)
 │ │ │ │ audio-1-1709554382234.aac (Aduio data)
 │ │ │ │ base.txt (Robot mobile base motion data)
 │ │ │ │ IMU-1.txt (IMU data)
 │ │ │ │ left_master_arm_joint-0.txt (Data of left master arm joint-0)
 │ │ │ │ left_master_gripper.txt (Data of left master gripper)
 │ │ │ │  left_slave_arm_joint-0.txt (Data of left slave arm joint-0)
 │ │ │ │ left_slave_gripper.txt (Data of left slave gripper)
 │ │ │ │ pan_tilt.txt (Data of pan tilt)
 │ │ │ │ right_master_arm_joint-5.txt (Data of right master arm joint-5)
 │ │ │ │ right_master_gripper.txt (Data of right master gripper)
 │ │ │ │ right_slave_arm_joint-5.txt (Data of right slave arm joint-5)
 │ │ │ │ right_slave_gripper.txt (Data of right slave gripper)
 │ │ │ │
 │ │ │
 │ │ │ ├─cam-1 (Images collected by camera 1. The camera sampling frame rate should be
 │ │ │ │  > =30FPS)
 │ │ │ │ 1709554382234.png
 │ │ │ │1709554383638.png
 │ │ │ │
 │ │ │ ├──cam-2
 │ │ │ │ 1709554382234.png
 │ │ │ │ 1709554383638.png
 │ │ │ │
 │ │ │ ├──lidar-1 (Point cloud collected by lidar 1, xyz unit: m)
 │ │ │ │ 1709554382234.ply
 │ │ │ │ 1709554382334.ply
 │ │ │ │
 │ │ │ ├──lidar-2
 │ │ │ │ 1709554382235.ply
 │ │ │ │ 1709554382354.ply
 │ │ │ │
 │ │ │ ├──rgbd-1 (Images and point clouds collected by rgbd1)
 │ │ │ │ 1709554382234.ply
 │ │ │ │ 1709554383630.ply
 │ │ │ │
 │ │ │ └──touch-1 (Data collected by touch sensor 1)
 │ │ │ │         1709554382234.txt
 │ │ │ │
 │ │ └──episode-2
 │ └──task-2
 │       │ description.yaml
 │       │ take_record.mp4
 │       │
 │       └──episode-1
 │
 └──series-2
   │ information.yaml
   │ AgileX Robot 2 instruction
   │
   └──task-1
     │ description.yaml
     │ take_record.mp4
     │
     └──episode-1
```

1.2 Data Sampling Plan

(1) **Scenario**. Data sampling scenarios should be diverse. Indoor scenarios such as： Bedroom, kitchen, shopping mall, dining room, cafe, living room, etc.， Outdoor： Park, residential district etc.

(2) **Action**. Robot actions should be diverse， such as： Pick、 move、 open、 close、 push、 place、 put、navigate、 separate、 point、 insert、 knock、 drag、 drop、 wipe、 assemble、 turn on etc.

(3) **Required**. Among the data in each mode, text instructions and images (including videos) are required for each collection. If it is an operation-related task, the robot end and gripper status data should also be collected. If it is a navigation-related task, the motion status data of the robot body should be collected, and other data should be collected as much as possible if conditions permit.

(4) **Default unit**. The timestamp of data recording is a Unix timestamp in ms. Each sensor should use the same timestamp reference. The default unit of angle is degrees.

(5) **Task**. For the same task, such as grasping an object, you can collect multiple episodes. You can place the object in different positions on the table in different postures, or even adjust the position of the robot body to collect corresponding episodes. It is recommended that the number of episodes collected for a task be no less than 10.

(6) **Video recording**. It is recommended that you use a mobile phone or camera in a third-person perspective to collect a task_record.mp4 for each task, and record the environment and operation information related to the task for easy understanding.

(7) **Camera data collection**. taking camera 1 as an example, in the cam-1 directory, each frame image is named with a timestamp and the format is png. All camera data mentioned in the description.yaml file must be collected. The acquisition frame rate is no less than 30 FPS.

(8) **Lidar point cloud data collection**, taking lidar 1 as an example. In the lidar-1 directory, each frame point cloud is named with a timestamp, the format is ply, and the xyz unit is m. All lidar data mentioned in the description.yaml file must be collected. The acquisition frequency is no less than 10 Hz.

(9) **rgbd camera data collection**, taking the rgbd 1 camera as an example, collecting images and point cloud data at the same time. In the rgbd-1 directory, each frame of image and point cloud is named with a timestamp. The image format is png, and the point cloud format is ply. The same frame images and point clouds should be consistent in timestamps. All rgbd data mentioned in the description.yaml file must be collected. The acquisition frame rate is no less than 30 FPS.

(10) **Touch sensor data collection.** Taking touch 1 as an example. In the touch-1 directory, each frame of data is named with a timestamp. All touch data mentioned in the description.yaml file must be collected.

(11) **The collection of body mobile data.** The collection of body mobile data is recorded in base.txt. The format is: Each line records once the collected data. Each line contains the timestamp, x, y, and heading in order. Each variable is separated by a space, and the rule of positive direction of the heading is to go from the positive x direction to the positive y direction. The unit of x and y is m. The acquisition frequency should preferably not be lower than 30 Hz.

(12) **The data collection of the arm end gripper.** The data collection of the arm end gripper is recorded in left_gripper.txt and right_gripper.txt respectively for the left and right arm. Only the right_gripper.txt is recorded for the single arm. The format is: Record the collected data once per

line, and each line contains in order the timestamp, x, y, z, roll, pitch, yaw, gripper opening and closing status, each variable is separated by a space. The units of x, y and z are m. When the gripper is open, it is 0, when it is closed, it is 1, and if there is an intermediate state, it is mapped to 0-1. The collection frequency is preferably no less than 30 Hz

(13) **Audio data.** For the collection of audio data, in the episode directory, as many audio files are collected as there are recording devices. The recording is continuously collected throughout the episode process. The file naming format is "audio-device serial number-initial timestamp.aac", such as: audio -1-1709554382234.aac.

(14) **Arm joint data.** In the description.yaml file, if recorded_left_joints and recorded_right_joints are not empty, the data of the corresponding joint should be collected. For example, left_joint-0.txt collects the data of joint No. 0 of the left arm. The format is: Recording the data collected once per line. Each line contains timestamp, motion value, and variables in order, separated by spaces. Other arm and joint data can be collected analogously. The acquisition frequency should preferably not be lower than 30 Hz.

(15) **Collection of head gimbal data.** In the information.yaml file, if pan_tilt is True, the data of the robot head gimbal should be collected, such as pan_tilt.txt, with the format of: each line records the collected data once, each line is a timestamp in sequence, and the rest are collected in strict accordance with the settings of pan_tilt_control in information.yaml, and each variable is separated by a space.

(16) **Collection of IMU data.** In the information.yaml file, if IMU_num is greater than 0, the data of IMU should be collected, such as IMU-1.txt, with the format of: each line records the collected data once, each line is a timestamp in sequence, acceleration x, acceleration y, acceleration z, angular velocity x, angular velocity y, angular velocity z, each variable is separated by a space, the unit of acceleration is m/s2, and the unit of angular velocity is degree/s. At the same time, the IMU data sheet, such as 'IMU.pdf', should be attached to the same directory as the 'information.yaml' file, and the robot direction corresponding to the positive xyz direction of the IMU should be described in the robot manual.

(17) **Data collection on humanoid robots or quadruped robot dogs.** The movement of the center of the body can be regarded as the base, and the data can be collected in the format of base.txt. Other data such as leg/waist can be collected in the format of the end effector or arm joint, and the data files can be named "left_foot.txt", "left_foot_joint-0.txt", etc.

(18) **The naming** of each data file and directory should strictly follow the format requirements.

(19) **Instructions** related to the robot platform can be attached to the collection (Each series uses the same robot) or series (Each series uses different robots) directory.

(20) In actual collection, you can collect multiple episodes with similar actions for the same series (Same scene and robot). For example, if you take an object, you can place the object in different postures on the table in different positions, or change to other objects, or even adjust the position of the robot body and collect corresponding episodes respectively. It is recommended that the number of episodes collected for one action is no less than 10 times.

1.3 Detailed explanation of configuration files

1.3.1 commit.yaml

- author_name: Name of the author. For example: ‘David’.
- work_organization: Author’s organization. For example: ’PCL’.
- author_email: Author’s email. For example: ‘yhlDavid@pcl.ac.cn’.
- role: Author’s role. For example： 'engineer'
- dataset_name: Dataset submitted at this time. For example: ‘agilex’.
- license: Data open source license. Should support commercial use if it's signed. For example： 'CC BY 4.0'， 'MIT license'， 'Unsure'
- PII_exclude: Whether the dataset excludes personally identifiable information. Must be True. Responsibility lies with the uploader
- thirdparty_consent: If the data set contains third-party data, please ensure that it is authorized by the third party. It must be True. The relevant responsibility lies with the uploader.
- healthy_content: The content of the data set should be healthy and cannot contain content involving unhealthy themes such as violence, drugs, abuse, etc. It must be True, and the relevant responsibilities lie with the uploader.

1.3.2 information.yaml

- series_name: The series name can be named according to the platform, scene, and agent form. For example ： 'Agilex_kitchen_bimanual'
- scene: Scene description: Indoor/Outdoor, kitchen/living room...For example ： 'indoor, kitchen'
- is_simulated: Real data or simulated data. True for simulated one and False for real one.
- morphology: Robot morphology. Multiple options. For example ： ['bimanual','wheeled']，

Range:

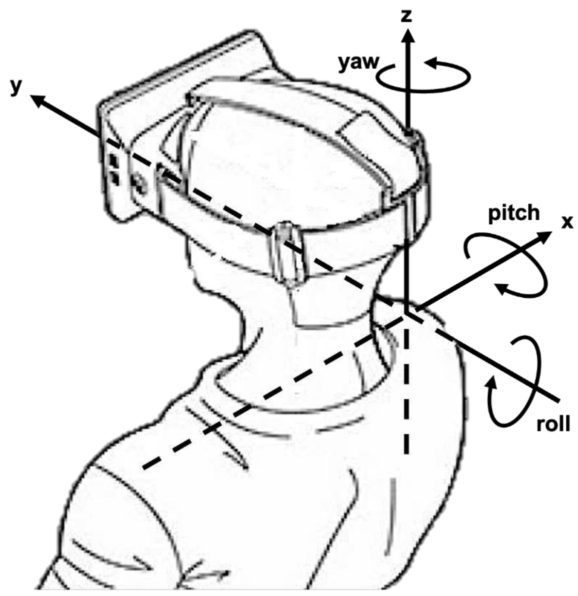
-'bimanual'： bimanual arms
-'single_arm'： single arm
-'AGV'： Automatic guided vehicle
-'quadrupedal'： quadrupedal robot
-'wheeled'： wheeled robot
-'drone'： drone
-'humanoid'： humanoid robot

- num_joints_per_arm: The number of joints of each arm, excluding the gripper, range >=0, fill in 0 if there is no arm.
- gripper: Is the gripper controlled in an open/closed binary state or a continuous state?

Range：

-'binary'
-'continuous'

- same_coordinate: For the collected actuator end data or body movement data, whether its coordinate system direction is the same as the preset coordinate system direction, if the same, True. The preset coordinate system is shown in the figure below. The right side of the robot is the positive x direction, the front is the positive y direction, and the top of the head is the positive z direction. The origin of the coordinate system is at the shoulder of the right arm joint, that is, the center of the connection between the right arm and the body. The robot coordinate system can be different from the preset coordinate system, but they must all follow the right-hand rule, and the correspondence between x-pitch, y-roll, and z-yaw should be consistent, that is, x cannot correspond to roll.



- endpoint_transform and origin_offset:

  If same_coordinate is False, you need to specify the rotation matrix and translation coordinates of the end effector's coordinate system to the preset coordinate system, such as endpoint_transform is [[1,0,0],[0,1,0],[0,0,1]], and origin_offset is[0,0,0]. The calculation formula is: collected end coordinates · endpoint_transform + origin_offset = end coordinates in the preset coordinate system. If the origin_offset cannot provide an accurate value (the error is less than 1cm), it can be omitted and set to: []. The robot without an arm is also: []. The direction of the end effector coordinate system should be attached in the robot manual.
- base_transform:

  If same_coordinate is False, you need to specify the transformation matrix from the coordinate system of the body movement to the preset coordinate system, such as: [[1,0,0], [0,1,0], [0,0,1]]. The calculation formula is: the collected body movement coordinates · base_transform = the body movement coordinates in the preset coordinate system. The robot manual should include the direction of the body movement coordinate system.
- IMU_transform:

  If same_coordinate is False, you need to specify the transformation matrix from the coordinate system of the IMU to the preset coordinate system, such as: [[1,0,0], [0,1,0], [0,0,1]]. The calculation formula is: the collected IMU coordinates · IMU_transform = the IMU coordinates in the preset coordinate system. The robot manual should include the direction of the IMU coordinate system.
- endpoint_control:

  The motion coordinate format of the actuator end. Select the data content to be collected from the following optional range. The data content and order of each line in left_gripper.txt and right_gripper.txt, except for the first timestamp and the last opening and closing state, should strictly correspond to the content of endpoint_control, such as ['absolute', 'x', 'y', 'z' ,'pitch', 'roll', 'yaw'], indicating that the end xyz and pitch, roll, yaw angles are to be recorded in absolute values.

Optional range: -'absolute'/'relative': absolute value/relative value, absolute value means that each collected motion data is based on the value of the preset coordinate system or other body coordinate system, relative value means that each collected motion data is the change value based on the previous motion. This concept is only for position and attitude.

If there is position or attitude data, one of 'absolute'/'relative' must be selected. It is not for speed, angular velocity, or torque.

- 'x', 'y', 'z': position coordinates
- 'pitch', 'roll', 'yaw': posture, 'pitch', 'roll', 'yaw' correspond toxyz axis rotation
-'vx', 'vy', 'vz': xyz direction movement speed
-'wx', 'wy', 'wz': xyz direction rotation speed
-'tx', 'ty', 'tz': xyz direction torque
-'none': no arm, or cannot move, or no relevant data is recorded

● base_control:

The robot body motion coordinate form, select the data content to be collected from the optional range. The data content and order of each line in base.txt, except for the first timestamp, should strictly correspond to the content of base_control, such as: ['relative', 'x', 'y', 'yaw'], indicating that the xy displacement and yaw rotation angle of the body should be recorded as relative values.

Another example: ['vx', 'vy','wz'], indicating that the speed in the xy direction and the angular velocity around z should be recorded. For the optional range, refer to endpoint_control.

● joint_control:

The form of joint motion of the robot arm. Select the data content to be collected from the optional range. In the joint data files such as left_arm_joint-0.txt, the data content and order of each line, except for the first timestamp, should strictly correspond to the content of joint_control, such as ['absolute', 'pitch', 'wx', 'tx'], which means that the angle, angular velocity, and torque of the joint should be recorded in absolute values. Since the joint has only one degree of freedom, the default is the x direction. For the optional range, refer to endpoint_control.

● pan_tilt: True if the robot head is a movable pan-tilt, otherwise False.

● pan_tilt_control:

If pan_tilt is True, you need to set the coordinate format of the head gimbal motion, and select the data content to be collected from the optional range.

The data content and order of each line in pan_tilt.txt, except for the first timestamp, should strictly correspond to the content of pan_tilt_control, such as: ['absolute', 'pitch', 'yaw'], indicating that the pitch and yaw angles of the gimbal should be recorded in absolute values. The coordinate system should follow the provisions of the preset coordinate system. For the optional range, refer to endpoint_control.

● endpoint_origin:

If the end effector uses absolute coordinates, this specifies the position of the coordinate origin on the robot. The optional range is:

-'shoulder': the root of the arm
-'middle': the middle of the robot body

-'head': the head of the robot

-'bottom': the bottom of the robot

-'none': non-absolute coordinates or no such information

- action_frequency: the robot control action frequency, unit: Hz, such as: 30
- blocking_control:

  whether it is blocking control, that is, the action of the current instruction must be completed before the next instruction can be executed. True indicates blocking control
- arm_num:

  the number of robot arms to be collected, such as: 4. If the number of arms exceeds 2, the control relationship between each arm and the human should be described in the robot manual.
- arm_operation_mode:

  robot arm operation mode, the list length should be equal to the number of arms, the order of elements is: [master left, master right, slave left, slave right, ...], such as: ['kinesthetic', 'kinesthetic', 'teleoperation', 'teleoperation'],

  elements optional range:

  -'kinesthetic': people directly move the robot to the specified position or push the robot to the specified position

  -'manipulation': people operate through remote control

  -'teleoperation': people have motion sensors on their hands or bodies to collect human motion data, and the robot moves according to the collected data

  -'imitation': the robot learns/imitates human motion through sensors such as laser radar/camera, and the robot has no direct contact with people
- camera_num:

  the number of cameras, how many cameras there are, the following needs to fill in the information of the number of cameras according to the example, such as: 2
- cam_view:

  camera installation viewing angle, the list length should be equal to the number of cameras, such as: ['ego-centric','third-person'], optional range:

  -'ego-centric': first person, generally refers to the camera installed on the top of the head

  -'third-person': third person, generally refers to the camera installed on the table or the surrounding environment

  -'left_wrist': camera installed on the left wrist of the robot or the end of the actuator

  - 'right_wrist': camera installed on the right wrist of the robot or the end of the actuator
- cam_calibration_file:

  camera calibration parameter file, the list length should be equal to the number of cameras, such as: ['calibration_1.yaml',''], the parameter file is placed in the same path as information.yaml

  , if there is no calibration file, it can be: ''
- lidar_num:

  number of laser radars, how many laser radars there are, the following is to fill in the number of laser radar information according to the example, such as: 2
- lidar_position:

  laser radar installation position, the list length should be equal to the number of radars, such

as:
['head', 'bottom'], optional range:
-'head': robot head
-'middle': robot middle
-'bottom': robot bottom

- cam1_lidar1_calibration_file:
  relative position calibration parameter file of camera 1 and lidar1, such as: 'calibration_cam1_lidar1.yaml', the file is placed in the same path as information.yaml
  , if not, you can fill in: ''. If there are relative position calibration files of other cameras or lidars, you can also fill in this format, such as: cam2_lidar2_calibration_file:
  'calibration_cam2_lidar2.yaml'.
- rgbd_num:
  the number of rgbd cameras, how many rgbd cameras there are, the following is to fill in the number of rgbd camera information according to the example, such as: 1
- rgbd_view:
  the rgbd camera installation angle, the list length should be equal to the number of cameras, such as: ['third-person'], optional range:
  -'ego-centric': first person, generally refers to the camera installed on the top of the head
  -'third-person': third person, generally refers to the camera installed on the table or the surrounding environment
  -'left_wrist': camera installed on the left arm wrist or the end of the actuator of the robot
  - 'right_wrist': camera installed on the right arm wrist or the end of the actuator of the robot
- touch_num:
  number of touch sensors. How many sensors are there? Fill in as much information as the example below, such as: 2
- touch_position:
  touch sensor installation position. The list length should be equal to the number of sensors, such as: ['left, gripper', 'right, gripper']
- touch_manual:
  touch sensor manual. If a touch sensor is used, the manual should be attached in the same directory as the "information.yaml" file, such as: 'touch.pdf'. If not, fill in: " ”
- recorded_left_master_arm_joints:
  the master left arm joint number for which motion data is to be collected. The motion data of the gripper is not included here. The number here should be consistent with the actual collected data. The larger the value, the closer the joint is to the end, such as: [0,1,2,3,4,5]. If joint information is not collected, it can be: [].
- recorded_left_slave_arm_joints:
  the slave left arm joint number for which motion data is to be collected. The details are the same as above. If there is only 1 left arm, this item is [].
- recorded_right_master_arm_joints:
  the master right arm joint number to collect motion data, the same as above, such as: [0,1,2,3,4,5].
- recorded_right_slave_arm_joints:
  the slave right arm joint number to collect motion data, the same as above, if there is only 1

right arm, this item is [].

- audio_num: the number of audio sensors, such as: 1.
- audio_frequency:
  the audio sampling frequency, the list length should be equal to the number of sensors, unit Hz, such as: [48000], if there is no audio, you can fill in: [0].
- IMU_num:
  the number of IMUs, how many IMUs there are, the following needs to fill in the information of how many IMUs according to the example, such as: 1.
- IMU_position:
  the IMU installation position, the list length should be equal to the IMU Quantity, such as: ['bottom'],
  Optional range:
  -'head': robot head
  -'middle': robot middle
  -'bottom': robot bottom

1.3.3 description.yaml

- instruction: The action instructions given to the robot. Should be consistent with the robot's actions and should be described in English. For example： 'pick up the apple from the table and put it into the bowl'
- instruction_CH: Chinese action instructions for the robot, the meaning should be consistent with the English instructions, can be left blank, such as: 'Pick up the apple from the table and put it in the bowl', or left blank: ''
- skills: The robot skills involved in instruction, which are the verbs, such as: ['pick', 'put']

1.4 Data set upload and submission

（1） The open-source data platform is OpenI, and its homepage is: https://openi.pcl.ac.cn/. You can refer to the help document: https://openi.pcl.ac.cn/docs/index.html#/ to create a project and upload a data set. The format of the data set package file must be .zip or tar.gz. The platform limits the size of a single data set file to no more than 200G. A maximum of 10 data set files can be uploaded to one project.

（2） The submitter should first register in the OpenI community. After successful registration, send the user’s name to yeh@pcl.ac.cn. Note: Join the ARIO organization. You will receive a reply email after successfully joining.

（3） After joining the ARIO organization, the submitter creates a project in the OpenI community, and the project path is selected under ARIO, as shown in the figure below.

（4） After the project is successfully created, you can upload the data set file under the project. At this time, the aforementioned collection directory and its containing files should be packaged and uploaded together.

（5） Attention: The uploader should ensure that the uploaded data does not contain personally identifiable information and does not contain content involving pornography, gambling and drugs. If a third party is involved, the third party should obtain authorization. The uploader shall be responsible for any infringement or illegal liability caused by data disclosure. Participation in the uploading of datasets for projects within the ARIO organization constitutes agreement to this provision.

1.5 The difference between our data format and Open X-Embodiment

Our format retains the original data content and saves the timestamp for each data item, which is convenient for users to sample and process at different time intervals. Users can also convert our data into Open X-Embodiment or other formats based on a simple conversion program. It is extendable and universal and compatible with more platforms. Our data also adds more modes such as point cloud and tactile, and can customize the data collection of multiple moving objects such as the end of the actuator, the body, and the joints of the arm. It can also customize the collection of multiple variables such as position, posture, speed, angular velocity, torque, etc., which can support more complex tasks and more flexible robot control.