Robot Model Specifications [Preliminary]

This is a preliminary document, next major update will be published on April 12th. The section describing the Model Constraints are expected to be exhaustive and not undergo substantial changes until the tournament. Guidelines for writing your models and details on the reviewing procedure will be completed. If you developed or are using a tool which makes the process easier for you, feel free to share it with the TC and we will include it in this document. In case you feel that your team requires elements which are not described in this document (e.g., specific contact properties between the robot feet and the ground), contact the TC as soon as possible and we will examine your request. The guidelines with respect to writing and reviewing models are still expected to receive major updates.

Introduction

This document aims at providing teams with clear specifications regarding what is allowed in their robot models but also advices on how to write their models and review the one provided by other teams.

The robot models for the competition will have to be written in the PROTO format. This format generally allows for a wider variety of sensors than what can be used for the Humanoid League, therefore we will present additional constraints in this document.

The main concerns of those rules and the reasons for these restrictions are the following:

- From a **research** point of view, this competition aims at providing an environment for transfer learning through sim2real experiments.
- In order to achieve **realism**, the characteristics and design of the robots should resemble robots used in regular competitions.
- To ensure a **fair** competition, teams who try to model their robots accurately by taking into account noise models and flaws of their hardware should not be penalized with respect to others.
- Using an **automatic referee** implies a few additional constraints on the robot models.
- Since the **resources** available for the simulation are limited, restrictions on the architecture of robots and on the accuracy of meshes may be required.

Model constraints

All the sensor and actuator nodes that are not explicitly mentioned in this section are forbidden during the competition.

Exported parameters

As the robot model must be in the PROTO syntax, the following fields have to be exposed:

- `translation`
- `rotation`
- `name`
- `controllerArgs`

All the exposed field are automatically filled by the referee. Other parameters may be exposed, but they will be ignored (e.g. set to their default value).

It is encouraged to expose key parameters of the model such as the parameters for each type of motor used.

```
PROTO MyRobot [
  field  SFVec3f     translation     0 0 0
  field  SFRotation  rotation        0 1 0 0
  field  SFString    name            ""
  field  MFString    controllerArgs  []
  field  SFFloat     MX64-torque     xxx
  field  SFFloat     MX64-vel        xxx
  field  SFFloat     MX64-damping    xxx
  field  SFFloat     MX64-backlash   xxx
  field  SFFloat     MX106-torque    xxx
  field  SFFloat     MX106-vel       xxx
```

```
  field  SFFloat     MX106-damping   xxx
  field  SFFloat     MX106-backlash  xxx
]
{
  Robot {
    translation IS translation
    rotation IS rotation
    name IS name
    controllerArgs IS controllerArgs
    ...
  }
}
```

As mentioned in the README provided by Cyberbotics, you should parse the `name` field and ensure your robot adapts to it to display its team color and player number:

```
if fields.name.value ~= '' then
  -- name is supposed to be something like "red player 2" or "blue player 1"
  local words = {}
  for word in fields.name.value:gmatch("%w+") do table.insert(words, word) end
  local color = words[1]
  local number = words[3]
```

Then, the `color` and `number` variables should be used by your PROTO file to display the requested color and player number. This can be achieved by forging a texture name from these variables or using them directly to assign material colors, create shapes, etc.

Sensors

For several sensors, a Look up Table (LUT) is used to specify the response of the sensor. This information is also used to specify the limits (min and max) for each sensor and the noise profile. This table should always be set according to the hardware specifications of the sensor you are modelling.

The following list of sensors is allowed with the restrictions mentioned here:

- Position sensors
  - The resolution of sensors should match the hardware specifications. For example, a 12 bit rotary encoder (like in the Dynamixel) has a resolution of $\frac{2\pi}{2^{12}} \approx 0.0015$.
- Accelerometer
  - LUT should be specified
- Gyro
  - LUT should be specified
- Touch Sensor
  - The 3 different options are allowed `Bumper`, `Force`, `Force-3d`.
  - Since the calculation of forces in the simulation is inherently noisy, no extra noise needs to be modeled for these sensors.
- Cameras
  - The maximal amount of raw data that a team is allowed to create through cameras is set to 100MB/s (MegaByte/second) *per team* (based on raw RGB images). This requirements ensures that the rendering of images is not slowing down the simulation. An example of valid configuration with 4 robots is the use of 640*480 images at 27 FPS.
    - Note: This limit is above the global network limit. Therefore, in order to use all of this bandwidth, you will need to use `JPEG` compression on the network which is not implemented yet.

Note: for the accelerometer and Gyro sensors, an offset drifting slowly will be added inside the simulator. This will model the fact that those sensors are highly sensitive to parameters such as the temperature, however those parameters will not be accessible in the PROTO file.

Actuators

Active joints can be implemented in two different ways:

- HingeJointWithBacklash for angular articulations, with the following child:
  - A RotationalMotor in the `device` field with the field `maxTorque` set to a value matching the hardware specifications. Please note that the `maxTorque` is not equivalent to the stall torque but must be read from the performance graph of the motor datasheet.
  - The `backlash` and `maxVelocity` parameter set according to the real hardware.
  - The `jointParameters` should be set according to the real hardware parameters.
    - `dampingConstant` should be set such that torque is zero at maximal velocity of the motor. Additional details for this will be provided in the next version.
    - `springConstant` if your robot has a Parallel Elastic Actuator (PEA) a spring force matching the real robots PEA may be specified.
- Hinge2JointWithBacklash for angular articulations around two axes. This is preferred when possible as it speeds up the simulation.
  - parameters as with the HingeJoint but for each motor individually
- SliderJoint for linear articulations, with the following child
  - A linear motor with the field `maxForce` set to a value matching the hardware specifications.
  - `JointParameters` should be set similar to the joint parameters of a HingeJoint

Both types of joints need to have the following children as well:

- A position sensor matching the constraints expressed in the previous section

Structure of the robot

Teams should provide in a dedicated file the configurations to be used:

- For upright posture (with fully extended knee)
- Longest extension posture

The format of the file should be as follows (values in rad):

```
{
    "team_name" : "MyTeamName",
    "upright" : {
        "motor_1_name": 0.123,
        "motor_2_name": 0.456,
        ...
    },
    "extension": {
        "motor_1_name": 0.123,
        "motor_2_name": 0.456,
        ...
    }
}
```

The robot should need to comport body parts with the following annotations as suffixes on the Solid names:

- `[foot]`: for all the body parts that can be in contact with the ground when walking. (e.g. `Solid.name "left foot [foot]"`)
- `[arm]`: for all body parts between shoulder and hand. Those parts are not allowed to touch the ball. (e.g. `Solid.name "left elbow [arm]"`)

Some joints (i.e. motors) should also be annotated with the following suffixes:

- `[shoulder]`: axis of the first joint of the arm (e.g. `name "right shoulder pitch [shoulder]"`).
- `[hip]`: for the first leg joint with an axis lying parallel to the ground plane (e.g. `name "left hip yaw [hip]"`).

Guidelines for writing your models

For general guidelines on how protos should be written please view the Cyberbotics documentation.

Usage of the `DEF/USE` mechanism as well as splitting up the `PROTO` into multiple subfiles (especially each visual or more complex collision model in its own file).

If you already have a `URDF` of your robot, using urdf2webots can give you a good starting point but manual adjustments are probably required.

Collision Model Complexity

Since simulation has to be performed at a reasonable real time factor, collision models (e.g., boundingObject) should be constructed from geometric primitives (Box, Capsule, Cylinder, Sphere).

To manually create these collision models we suggest a tool based on onshape-to-robots pure shape approximation which will be released in the near future.

If you are already using a `URDF` model of your robot, rotated bounding boxes can be generated using the `simplify_urdf_collision` tool.

Visual Model

While visual model complexity (i.e., the number of triangles) is a lesser issue than collision model complexity, it should still be set reasonably low (i.e. below 50 000 vertices in total).

When exporting a model from a CAD software, the resolution of the mesh approximation can usually be set. When only a mesh model is available, sofware such as Meshlab's Quadratic Edge Collapse Decimation.

We suggest to apply an Appearance or PBRAppearance matching the real robot. Several appearances are provided by Cyberbotics and described here.

Model inspections

Semi-automated validation

The following properties of the robot are extracted automatically:

- `Htop`: In upright posture, the amplitude along the z component
- `Hleg`: In upright posture, z component between `[hip]` and the minimal value of the robot along z axis
- `Hhead`: In upright posture, z component between `[shoulder]` and the maximum value along z axis.
- `M`: the total mass of the robot
- `Hcom`: In upright posture, the height of center of mass.
- `BMI = M/Htop^2`
- `Width`: In upright posture, the diameter of the smallest cylinder in which the robot can fit
- `FootWidth, FootLength`: The size along y (resp x) axis of the bounding box for one foot.
- `MaxLength`: The maximal distance between two points of the robot in longest extension posture.

The following constraints are checked automatically

- `5 <= BMI <= 30`
- `(FootWidth*FootHeight) <= (2.2*Hcom)^2/32`
- `1.2 <= max(FootWidth,FootHeight)/min(FootWidth, FootHeight) <= 3.5`
- `Width <= 0.55 Htop`
- `0.35 Htop <= Hleg <= 0.7 Htop`
- `0.1 Htop <= Hhead <= 0.3 Htop`
- Check that only allowed sensors/actuators are used
  - All required fields should also be set (e.g., LUT)
- are only geometric primitives used as collision model?
- complexity of visual model?

A document containing the following information is generated from the robot model:

- List of the properties of the robot model extracted automatically

- List of all the joints with all parameters specified in the respective section
- List of the Sensors and their Parameters
- Kinematic chain/loop of the robot ??
- TO BE COMPLETED

Peer-reviewed validation

The points that should be validated during the review process are the following:

- Are specifications for custom actuators/sensors valid
- Visual check of annotations: `[foot]`, `[arm]`, `[shoulder]`, `[hip]`
- TO BE COMPLETED

TC validation

In order to ensure reasonable performance of the simulation, teams will be required to upload a simple docker image with a walking behavior. It should be possible to simulate two full teams using those robots on the hardware used for the competition. In case the robot is too complex teams will be contacted to take appropriate measures to reduce the complexity of the simulation.