Data structure for World Model sharing in MSL

Rev date: 2016/03/04

1. Introduction

The need for ever growing data standardization among the MSL community is an important step towards the ability of extracting practical and real data for debugging, statistical analyses and benchmarking during tournaments, as well as providing an effective way of allowing easy ad-hoc mixed teams formation in the future.
With this in mind, it has been agreed upon, during the 2015 MSL Workshop held in Aveiro, Portugal, to establish the following data structure to be used as a standard by every MSL team, therefore allowing useful data logging during the games. This data may in the future be combined with a ground truth system, whenever a reliable one is made available to the MSL competitions. It was been common agreed that this approach is feasible in a short term and may already provide the teams with very useful information towards research evolution in the league starting in 2016 events.

2. Data Type and coordinate system definitions

All data regarding robots, obstacles or ball are to be defined in a world three dimensional Cartesian coordinate system defined according to the following rules (see figure 1):

- for any team, the World coordinate system is defined as viewed from the own side of the field;
- the World coordinate system follows the right system principle;
- the *'XY'* plane is coincident with the field floor;
- the origin of the coordinate system is coincident with the center of the field;
- the *'X'* axis is coincident with the center of the mid-field line and positive to the right;
- the *'Y'* axis is positive towards the opponent goal;
- the 'z' axis is positive above the ground;
- angles are measured from the *'X'* axis, or from any other line parallel to the *'X'* axis, in the counter clockwise direction, and are defined in the interval [0 .. 2π[;
- angle arithmetic uses (mod 2π).

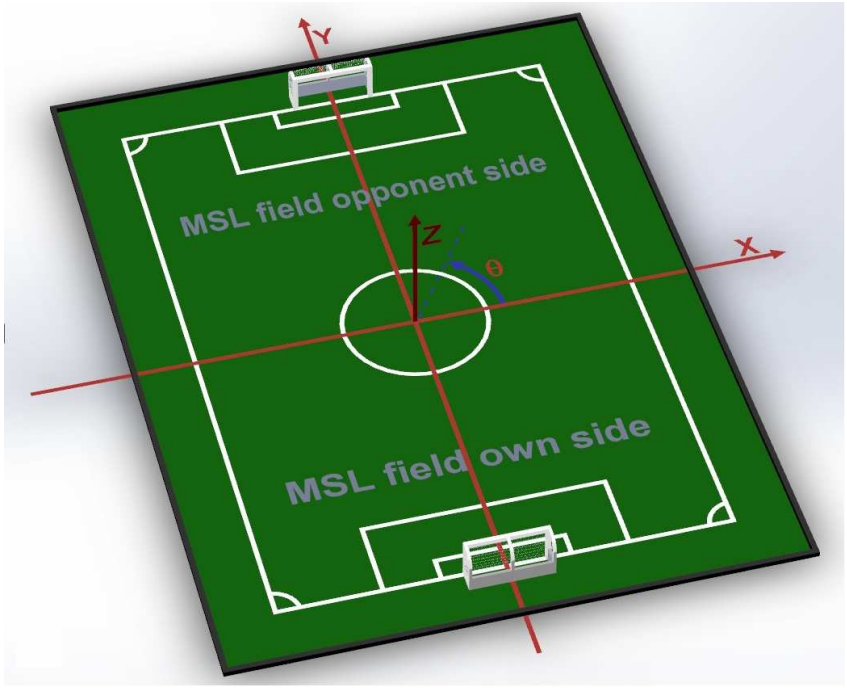

Fig. 1 - Three dimensional Cartesian coordinate system for world coordinate representation

Since, for any team, the coordinate system is defined as viewed from the own side of the field, to obtain any coordinates from the opponent team data structure a transformation must be performed.

This transformation can be expressed as

$$[x' \quad y' \quad z'] = [x \quad y \quad z] \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{and} \quad \theta' = \theta + \pi (\text{mod}\, 2\pi)$$

Besides the above rules the following ones also apply:

- positions are always defined as a triplet [x, y, z] in the above defined three dimensional Cartesian coordinate system;
- positions are defined in meters, with a minimum of three decimal digits of resolution;
- angles are defined in radians, with a minimum of four decimal digits of resolution;
- linear velocities are defined in *m/s*, as a vector starting at the coordinate system origin and defined by its cartesian representation $\mathbf{v_i} = [x_i, y_i, z_i]$.
- angular velocities are defined in *rad/s,* being positive in the direct direction (counter clockwise) and negative otherwise.
- velocity of a given object (e.g. the ball) is obtained by the translation of origin of the aforementioned vector to the object position coordinates at a given instant.
- robot heading is represented as an angle between the 'x' axis on the robot frame and the *'X'* axis in World Coordinates (see figure 2).

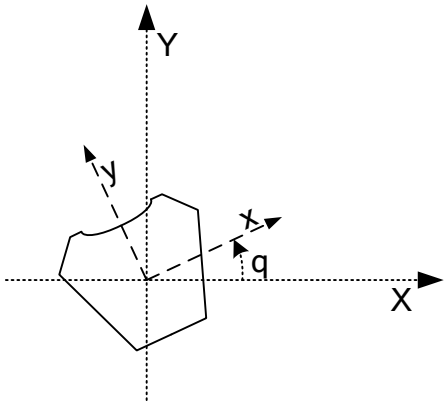

Fig 2 – Robot heading measured in the *xy* plane translated to the frame of the robot

3. Information to be provided

Each base station is required to send, at a periodic rate (minimum 10 times per second), and using the same tcp connection used to receive commands from the RefBox, a data structure containing the following information:

- A pre-defined "type" label. This will currently be "worldstate";
- The team name;
- A free format string, in English, with the *Current Intention* of the team (e.g. "Searching for the ball")
- Current ball position in the coordinate system;
- A list of robots. For each of the team five robots:
    - Robot id (numeric);
    - Current pose;
    - Target pose;
    - Current velocity;
    - A free format string with the *Current Intention* of the robot (e.g. "pass the ball")
    - Battery level expressed in percentage;
    - Boolean status regarding ball engage condition [0, 1];
- A list of balls, organized by confidence level. For each ball:
    - Current position;
    - Current velocity;
    - Degree of confidence;
- A list of obstacles. For each obstacle:
    - Current position;
    - Current velocity;
    - Estimated radius;
    - Degree of confidence;
- Information age in milliseconds. This information is used by the refbox tocalculate and LOG the timestamp;

Besides the "worldstate" type messages, teams can at any time send to the RefBox an "event" type message marking particular situations. The data structure of these messages should contain the following information:

- A pre-defined "type" label. This will currently be "event";
- The robot id generating the event (numeric);
- A free format string, in English, describing the event (e.g. "kick", "pass", ...)

In the "worldstate" type messages, and apart from the two first fields, which are mandatory, all other fields may be not defined or partially not defined depending on the ability of the team to determine them. The rules for constructing this structure, including missing data, is defined in the next section.
If the RefBox fails to receive the information from one or both of the base stations after a period of 100ms, it will fill in an empty data structure and store it in the logging file.
The log file will be publicly released to all teams at the end of each game.

4. Data structure formal definition

The data structure provided by each base station will follow the JSON (JavaScript Object Notation) notation. Detailed information about the grammar of this lightweight data-interchange format may be found in http://www.json.org/.
All information that composes the data structure (which will be an object in JSON) is text based, and its embedded structuring using simple markers such as "{" and "}" results in a light and easy to parse and/or read data-interchange packet. A large set of "off-the-shelf" tools for building or parsing JSON structures is also readily available for a large set of different programming languages.

The format of the MSL-JSON object instances to be delivered by each base station will have the structure depicted in the following pages:

MSL World Model data structure format:

```
{
    "name" : "MSLWMD",
    "description":"MSL World Model data",
    "type": "object",
    "properties":{
       "type":{  "type":"string",
                    "description":"must be worldstate"
       },
       "teamName":{"type":"string",
                    "description":"name of the team"
       },
       "intention":{"type":"string",
                    "description":"team intention in English e.g. [attack] "
       },
       "robots":{
          "type": "array",
          "items": {
             "type":"object",
                "properties":{
                   "id":  { "type": "number",
                      "description": "robot number"
                   },
                   "pose":  {"type": "array",
                      "items": {
                         "type":["number", "null"],
                           "description":"robot x position",
                         "type":["number", "null"],
                           "description":"robot y position",
                         "type":["number", "null"],
                           "description":"robot heading "
                      }
                   },
                   "targetPose": {"type": "array",
                      "items": {
                         "type":["number", "null"],
                           "description":"robot target x position",
                         "type":["number", "null"],
                           "description":"robot target y position",
                         "type":["number", "null"],
                           "description":"robot target heading (teta)"
                      }
                   },
                   "velocity":{"type": "array",
                      "items": {
                         "type":["number", "null"],
                           "description":"robot x velocity component in m/s",
                         "type":["number", "null"],
                           "description":"robot y velocity component in m/s",
                         "type":["number", "null"],
                           "description":"robot angular velocity in rad/s"
                      }
                   },
                   "intention":  { "type": "string",
                      "description": "robot intention, in English, e.g.(pass the ball)"
                   },
                   "batteryLevel":  { "type": "number",
                      "description": "battery level in percentage, e.g. 95.5"
                   },
                   "ballEngaged":   { "type": "number",
                      "description": "ball engaged status, either 0 or 1"
                   }
                }
          }
       },
       "balls":{
          "type": "array",
          "items": {
             "type":"object",
```

```
                "properties":{
                   "position": {"type": "array",
                      "items": {
                         "type":["number", "null"],
                           "description":"Ball x position",
                         "type":["number", "null"],
                           "description":"Ball y position",
                         "type":["number", "null"],
                           "description":"Ball z position"
                      }
                   },
                   "velocity":{"type": "array",
                      "items": {
                         "type":["number", "null"],
                           "description":"Ball x velocity component in m/s",
                         "type":["number", "null"],
                           "description":"Ball y velocity component in m/s",
                         "type":["number", "null"],
                           "description":"Ball z velocity component in m/s"
                      }
                   },
                   "confidence": { "type": "number",
                      "description": "confidence level [0, 1]"
                   }
                }
          }
       },
       "obstacles":{
          "type": "array",
          "items": {
             "type":"object",
                "properties":{
                   "position": {"type": "array",
                      "items": {
                         "type":["number", "null"],
                           "description":"Obstacle x position",
                         "type":["number", "null"],
                           "description":"Obstacle y position"
                      }
                   },
                   "velocity":{"type": "array",
                      "items": {
                         "type":["number", "null"],
                           "description":"obstacle x velocity component in m/s",
                         "type":["number", "null"],
                           "description":"obstacle y velocity component in m/s"
                      }
                   },
                   "radius":   { "type": "number",
                      "description": "obstacle radius in meters"
                   },
                   "confidence": { "type": "number",
                      "description": "confidence level [0, 1]"
                   }
                }
          }
       },
       "ageMs":{ "type":"number",
                   "description":"age of information in ms"
       }
    }
}
```

MSL game event data structure format:

```
{
    "name" : "MSLEVENT",
    "description":"MSL game event",
    "type": "object",
    "properties":{
       "type":{  "type":"string",
                    "description":"must be event"
       },
       "robotId":{"type":"number",
                    "description":"robot number"
       },
       "event":  {"type":"string",
                    "description":"type of event in English e.g. [kick, pass, …] "
       }
    }
}
```

5. Examples of each of the two JSON structures


```
{
  "type": "worldstate",
  "teamName": "MYTEAM",
  "intention": "Searching for the ball",
  "robots": [
    {
      "id": 1,
      "pose": [0.002, -8.933, 6.2800],
      "targetPose": [0.002, -2.993, 0.4934],
      "velocity": [0, -0.004, 0.0000 ],
      "intention": "",
      "batteryLevel": 93.0,
      "ballEngaged": 1
    },{
      "id": 2,
      "pose": [3.100, 2.345, 2.1200],
      "targetPose": [5.002, 4.300, 0.800],
      "velocity": [0.455, 0.650, 0.0200],
      "intention": "target receive",
      "batteryLevel": 95.0,
      "ballEngaged": 0
    },
   (...)
  ],
  "balls": [
    {
      "position": [-0.002, -0.001, 0],
      "velocity": [0, 0, 0],
      "confidence": 0.959
    },
    {
      "position": [null, null, null],
      "velocity": [null, null, null],
      "confidence": 0
    }
  ],
  "obstacles": [
    {
      "position": [-0.002, -0.001],
      "velocity": [0, 0],
      "radius": 0.25,
      "confidence": 0.959
    },
    {
      "position": [-3.150, -4.677],
      "velocity": [0, 0],
      "radius": 0.25,
      "confidence": 0.800
    },
   (...)
  ],
  "ageMs": 20
}
```

```
{
  "type": "event",
  "robotId": 2,
  "event": "kick"
}
```

5. Contributors

Bernardo Cunha
Ricardo Dias
Artur Pereira
Robin Soetens
Andreas Witsch
Andre Pool
Fernando Ribeiro
Filipe Amaral
Helder Ribeiro
Jan Feitsma
José Luis Azevedo
Koen Meessen
Lotte de Koning
Matthias Briegel
Nugroho Fredivianus
Nuno Braga
Pedro Osório
Ricardo João
Stefan Jakob
Stephan Opfer
Tiago Maia
Tim Kouters
Wouter Kuijpers