Attentional Trees in Robotics: towards an optimal executive attentional control meeting high-level decision making and control layer features.

Pilar de la Cruz

December 5, 2019

Abstract

Introduction

Behavior Trees Pros

- Ticking - the ability to tick allows for work between executions without multi-threading
- Priority Handling - switching mechansims that allow higher priority interruptions is very natural
- Simplicity - very few core components, making it easy for designers to work with it
- Dynamic - change the graph on the fly, between ticks or from parent behaviours themselves

JSON Schema

JSON is a standard format for data exchange inspired by JavaScript. Generally, JSON is in string or text format. A typical JSON schema consists of a list of string objects. The first object element defines the root of the Behavior Tree. The remaining object elements are defined as subtrees and will be added recursively as children from the root. Each object contains properties `name`, `type`, `father`, `children` and `parameters`, which provide with information about the Behavior Tree node types as well as additional parameters for pre-and post-conditioning handling, priority execution and any additional information. This schema provides (1) simplicity due to its compact description of information, (2) flexibility due to its abstract base representation of tasks and (3) modularity due to the definition of subtrees which can be added and/orremoved on the fly in whatever position of Behavior Tree.

```
[
  {
    "name":"sequence01",
    "type":"sequence",
    "father": "",
    "children": [],
    "parameters": []
  },
  {
    "name":"selector01",
    "type":"selector",
    "father": "sequence01",
    "children": ["isReached","reach"],
    "parameters": ["distanceToObject", "emphasis", "reached"]
  },
  {
    "name":"selector02",
    "type":"selector",
    "father": "sequence01",
    "children": ["isPicked","pick"],
    "parameters": ["picked", "emphasis"]
  },
```

```
  {
    "name":"selector03",
    "type":"selector",
    "father": "sequence01",
    "children": ["isPlaced","place"],
    "parameters": ["placed", "emphasis"]
  }

]
```

System Overview

Attentional Trees framework aims at integrating the following features:

1. JSON schema parsing and Behavior Tree integration
2. Task planner based on the creation of the Behavior Tree
3. Priority handling of tasks based on the emphasis value
4. Event-based approach of Behavior Tree tick execution

JSON Parsing

Task planner

Priority Handling

Event-based Behavior Tree

Implementation

The framework platform of Attentional Trees is based on the open-source py_trees library. Py_trees is compatible with ROS and provides the easiness and quickness of an implementation based in python language. Moreover, decoding operations becomes specially intuitive thanks to the in-built mapping between container and value (JSON objects to Python objects and viceversa) based on Python standard libraries.

Simulation Experiments

In order to test the requirements described in section above, the following demo programs have been implemented[1]

- `json-tree.py`. This program focuses on (1) loading JSON schema and (2) mapping into a Behavior Tree based on PA-BT approach.
- `emphasis-tree.py`. This program focuses on high-priority handling based on access to emphasis value[2] (add note, page).
- `attentional-tree.py`. This program focuses on event-based task execution of Behavior Trees. Tick signals are not periodically sent, but triggered by special events handled in the control layer.

0.1 JSON and Behavior Trees

References

[1] Add Matteo's paper and page of emphasis description

[2] https://www.guru99.com/python-json.html

[1] These are available on: