Computer-aided design of complex configurations and behaviors for modular robots

Author Names Omitted for Anonymous Review. Paper-ID [add your ID here]

***Abstract*—In this paper, we present a scalable software framework for the design of modular robot configurations and behaviors. Designs are constructed hierarchically by composing elements from a library, allowing users to easily create complex designs. Likewise, complex behaviors are constructed by composing controllers from a library in a nested series/parallel structure. The system is integrated with a full dynamic simulator, and provides tools to identify common problems with behaviors, specifically self-collision and loss of quasi-static stability.**

I. Introduction

- Introduce existing designs on modular robots.
- Introduce existing works on modular robot controller design
- Introduce SMORES modular robot and advantages
- Introduce the contribution of this paper
- Why fully automomous approach does not work well so far

II. Preliminary

a) SMORES robot module: Define the ability of motion and connectivity of a SMORES robot module. Position and velocity of each Dof. Can be connected to four other modules at the same time. Representation of properties of a SMORES module, e.g. joint angels, global positions, connection information

b) Configuration: Define the representation of a configuration as a set of SMORES robot modules connected in a certain way. Define topology graph.

c) Controller: Define controller as a basic feedback controller for each Dof of each SMORES module. The reference input of the controller is a gait table. Define how the gait table is executed. Mention that controller in this paper refers to the input gait table.

d) Collision: Define a collision between SMORES modules.

e) Controller conflict: Define a conflict between controllers, i.e. giving opposite commands to the same Dof of a module at the same time.

f) Unexpected behavior: Define the unexpected behavior of a configuration due to instability during a controller execution.

III. Approach and Algorithm

g) Configuration composition: Define the composition of a set of configurations to a single configuration.

A. Configuration Composition

h) Input: A set of configurations. A topology graph representing the connectivity among those configurations. A base module (for position transformation).

i) Output: A composed configuration if it is safe.

j) Procedure:

- Start from the configurations that connects to the configuration with base module, transform their positions based on the position of the base configuration and topology graph.
- Check if there is any collisions among the modules and report such collision.
- **Check if the final configuration is stable. If not, find the plane that will make the configuration stable and transform the configuration.**
- Show the expected behavior in simulator.

k) Controller composition: Define the composition of a set of controllers to a single controller. Define the difference between a parallel composition and a series composition. Define the control composition graph.

B. Controller Composition

l) Input: A configurations. A set of controllers. A control composition graph.

m) Output: A composed controller if it is safe.

n) Procedure:

- Compose the set of controllers based on the given control composition graph. Explain how the parallel composition and series composition are handled.
- **Check there is no controller conflict in the composition.**
- Execute the composed controller in user defined incremental time interval. At each time step, update each module position and check collision.
- **At each time step, check if the configuration will not have any unexpected behavior.**

C. Complexity

Discuss the complexity of the algorithm with respect to the number of modules and size of gait tables.

IV. Example and Experiment

With simulation in Gazebo:

- Show a configuration composed from a set of basic configurations.
- Show a composed controller that results in a collision in the configuration.

- Show an updated controller that resolves the collision
- Show a composed controller that results in an unexpected behavior.
- Show an updated controller that eliminates the unexpected behavior.

V. Conclusions

We worked hard, and had fun.

VI. Future

- How to represent different attribute/ability of the configurations

References