**Topic**

I plan to investigate the viability of integrating modularity into continuous self-modeling in the task of hexapedal locomotion. I hypothesize that such an approach would exhibit promising performance in terms of damage recovery.

The robot will be evolved to move towards a light source. Both a simulator and a physical hexapedal robot will be used in this project. The simulator will be used to evolve the robot's controller, and itself will also be subject to online evolution, with the evolutionary goal of predicting proprioceptive data close to those from the physical robot (Bongard et al., 2006). The controller will be a structurally and functionally modular Continuous Time Recurrent Neural Network (CTRNN). The connections will be defined pre-evolution in a way that is structurally modular, with one local network for each leg, and another overarching network that takes all sensor values as input and reaches out to each aforementioned local leg network. The intention of this design is to have one CPG to control the gait and one local network to control each leg; therefore the controller is expected to be not only structurally modular, but also functionally modular. In order to establish the overarching network as the CPG, neurons in the overarching network will be defined with high $\tau$ values (i.e. their activations will remain relatively constant), whereas neurons in the local networks will be defined with low $\tau$ values (Yamashita & Tani, 2008). Please see Appendix for a preliminary architectural design of the network.

The controller should be first evolved with the undamaged robot using reality-gap-bridging techniques such as introducing noise (Jakobi, 1997) and multi-objective optimization with transferability (Koos et. al., 2013). After reasonable fitness has been achieved, the controller will be transferred onto the physical robot together with the simulator (which may either be hand-crafted or evolved by self-modeling; the simulator doesn't have to be perfect at the start). When damage occurs, the robot will evolve its simulator by self-modeling (in short, do test moves and see how the data returned from the physical proprioceptive sensor compare with those from the simulator's prediction), and then use the simulator to evolve the controller (the entire network will be evolved as a whole).

## Rationale

Adaptive robots that are capable of recovering from damage have significant application prospectives in environments that prefer minimal human intervention. One approach to implementing this feature was to maintain a reservoir of controllers, each built to deal with a specific scenario, and to select from this reservoir by trial and error when failure occurs (Cully, et. al., 2015; Chatzilygeroudis, et. al., 2018). There had also been an attempt to build the robot out of soft material and reshape the robot to make up for the damaged parts (Kriegman, 2019). Yet a third approach, which I took inspiration from, was to maintain online an evolvable simulator that frequently updates to approximate the physical robot; thus, when the robot is damaged, the simulator will update to match the robot’s new body scheme, and a new controller can be trained from the simulator (Bongard et al., 2006). In this way, the robot could obtain basic prospection, and thus adapt to novel situations with few physical trial-and-errors (Bongard, 2015). However, the original implementation uses a non-modular neural network which may be difficult to re-train.

A non-modular network is characterized by the dense and evenly distributed connections among its neurons. Therefore any function that a non-modular network implements will be distributed across the entire network. This would especially be a problem in our case because the performance of the network, whether good or bad, cannot be attributed to individual connections, and thus the connection to be mutated has to be chosen by random (for comparison, consider gradient descent in supervised training). Since functions are distributed across the entire network, any random mutation will cause disturbances across all functions the network implements. As a result, the successful components in the pre-mutation network can easily be lost after a few mutations.

By contrast, any single mutation on a modular network would only have limited impact on a subset of the network, and therefore the successful components in the pre-mutation network can be more easily preserved (Wagner & Altenberg, 1996). A network can be modular structurally and/or functionally. Structural modularity is when the network consists of modules that are densely connected within but sparsely connected in between.

Functional modularity is broadly defined as the principle that complex entities may be segmented into simpler elements and that simple elements may be integrated into a complex entity. For example, in our network, structural modularity is embodied as the distinction among each of the local leg networks and the overarching network, whereas functional modularity takes the form of the division of task into a hierarchical structure, from gait control to leg control. Thus, it is expected that when the controller network needs to be retrained, the mutations will not as easily destroy the existing functionalities but rather build on top of them, resulting in quick re-adaptation.

## Check Points

**10/13/2019**
Complete the introduction section of the thesis

**11/13/2019**
Finalize on the neural network architecture, robot body scheme, and other details (except for parameters such as mutation rate) about robot's design

**12/13/2019**
Complete simulator and physical robot control codes

**2/13/2020**
Complete self-modeling codes

**3/13/2020**
Obtain a controller that performs reasonably well on undamaged robot, and experiment with damaged robot

**4/17/2020**
Due

## Bibliography

Bongard, J., Zykov, V. & Lipson, H. (2006). Resilient Machines Through Continuous Self-Modeling. *Science* 314: 1118–1121.

Bongard, J. (2015). Using robots to investigate the evolution of adaptive behavior. *Curr. Opin. Behav. Sci.* 6:168–173. doi: 10.1016/j.cobeha.2015.11.008.

Chatzilygeroudis, K., Vassiliades, V. & Mouret, J. B. (2018). Reset-free Trial-and-Error Learning for Robot Damage Recovery. *Robotics and Autonomous Systems*. 100: 236-250. doi: https://doi.org/10.1016/j.robot.2017.11.010

Cully, A., Clune, J., Tarapore, D. & Mouret, J. B. (2015). Robots that can adapt like animals. *Nature* 521: 503-507.

Jakobi, N. (1997). Evolutionary robotics and the radical envelope-of-noise hypothesis. Adaptive Behavior, 6(2): 325–368.

Koos, S., Mouret, J.-B., and Doncieux, S. (2013). The transferability approach: Crossing the reality gap in evolutionary robotics. IEEE Transactions on Evolutionary Computation, 17(1): 122–145.

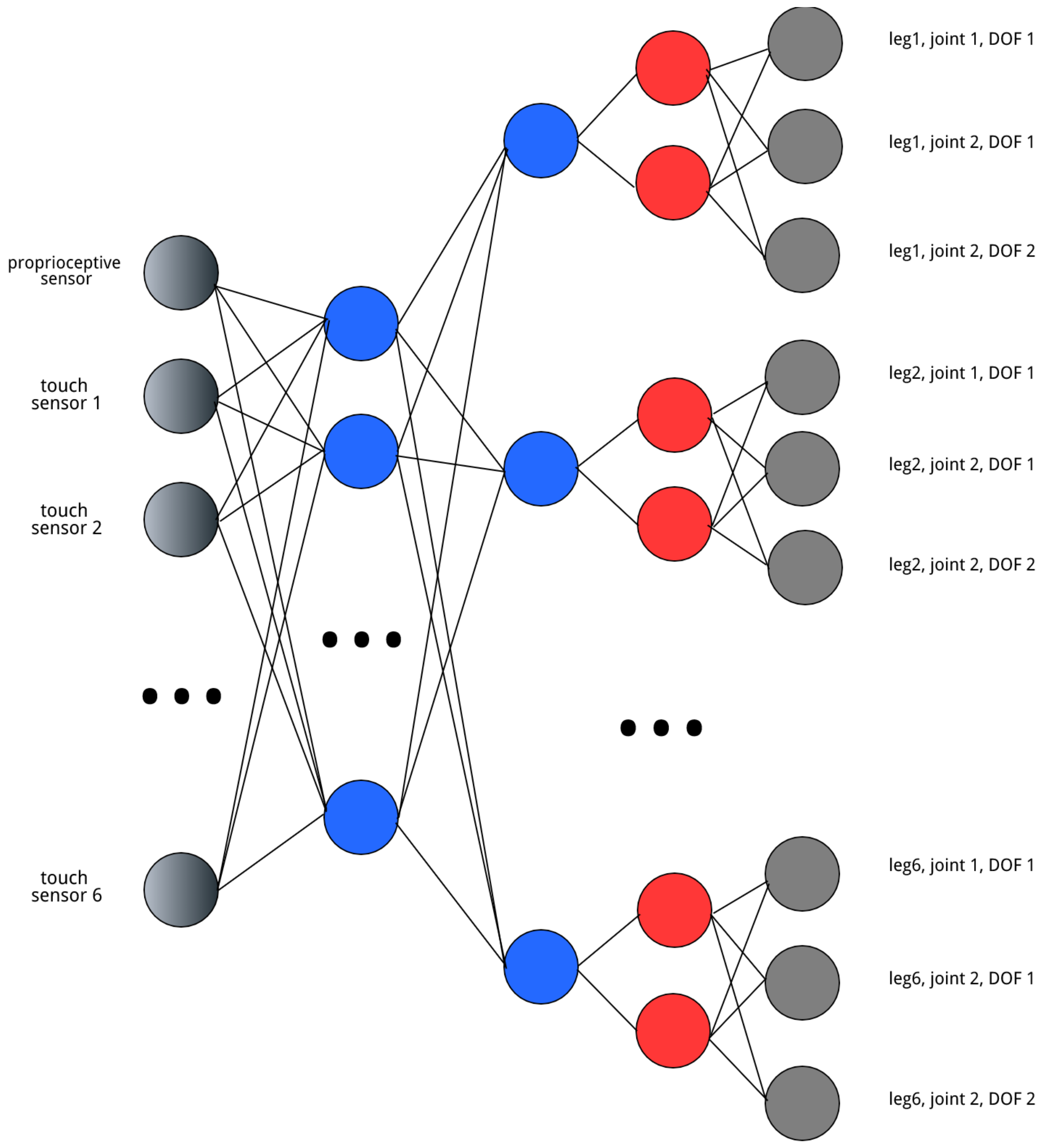
Kriegman, S., Walker, S., Shah, D., Levin, M., Kramer-Bottiglio, R., Bongard, J. (2019). Automated shapeshifting for function recovery in damaged robots. *Robotics: Science and Systems*.

Wagner, G. P. & Altenberg, L. (1996). Perspective: Complex adaptations and the evolution of evolvability. *Evolution 50(3)*: 967–976.

Yamashita, Y. & Tani, J. (2008). Emergence of functional hierarchy in a multiple timescale neural network model: a humanoid robot experiment. *PLoS Comput. Biol.* 4: 1–18.

## Appendix

This is a preliminary design of the controller architecture. Currently, the robot is designed to have a proprioceptive sensor and a touch sensor on each of its 6 legs; each of its 6 legs is designed to have a 1-DOF joint and a 2-DOF joint. Therefore, there are currently 7 input



neurons and 18 output neurons across the entire network. The dimensions may change if the current design proves to be too complex. The red hidden neurons are “fast” neurons intended to control individual legs, and the blue hidden neurons are “slow” neurons intended to control gait.