Cornell Ranger Robot Control

An energy-efficient strategy for bipedal robots

Executive Summary

- Use a high-level global state machine to dictate whether the controller code runs corrective actions or nominal actions
- Design controller code based on the behavior tree representing the modes of operation
- Develop modules for the abstracted behaviors and their low level commands
- Reuse modules to support additional modes of operation and future projects
- Save energy by performing checks on fewer states

State Machine Approach

- Problem: Using state machines to manage all levels of actions results in too many states
- Answer: Use a high-level state machine to represent the global status of the robot. Limit the checking done in global states.
  - Controller looks at the state machine for cues to take corrective actions or nominal actions

Global States

- Ranger is always in one of these states:
  - Stationary
  - Walking
  - Falling

Global State Transitions

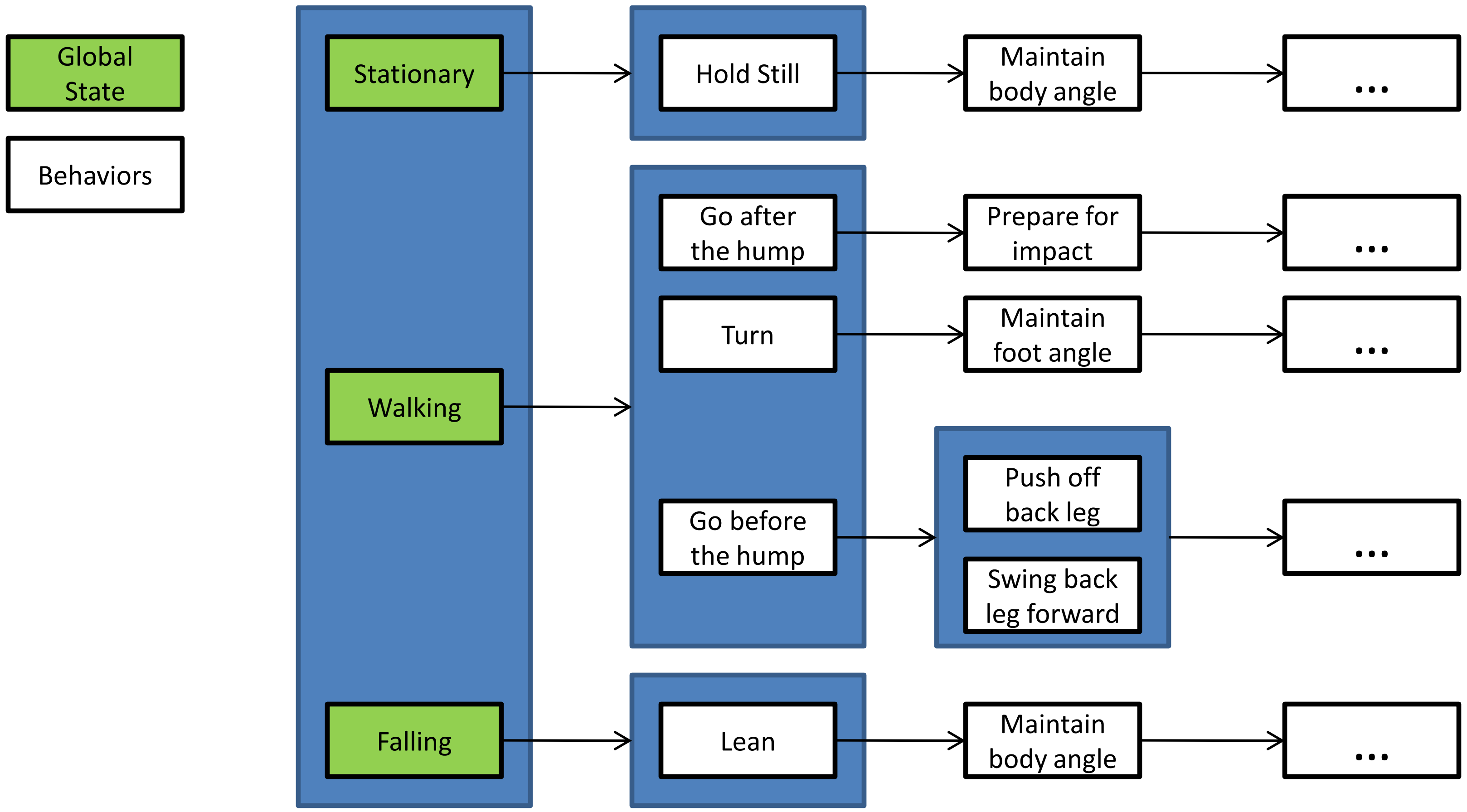
- Global states change on key observations
- Example: “Falling”
  - Observed unexpected sign change in velocity (falling backwards)
  - Observed greater than expected velocity values (falling forwards)

Modes of Operation

- Ranger has these modes of operation:
  - Stand
  - Walk forward
  - Walk backward
  - Turn
  - Any sequence of the above
- Other possibilities: Dancing

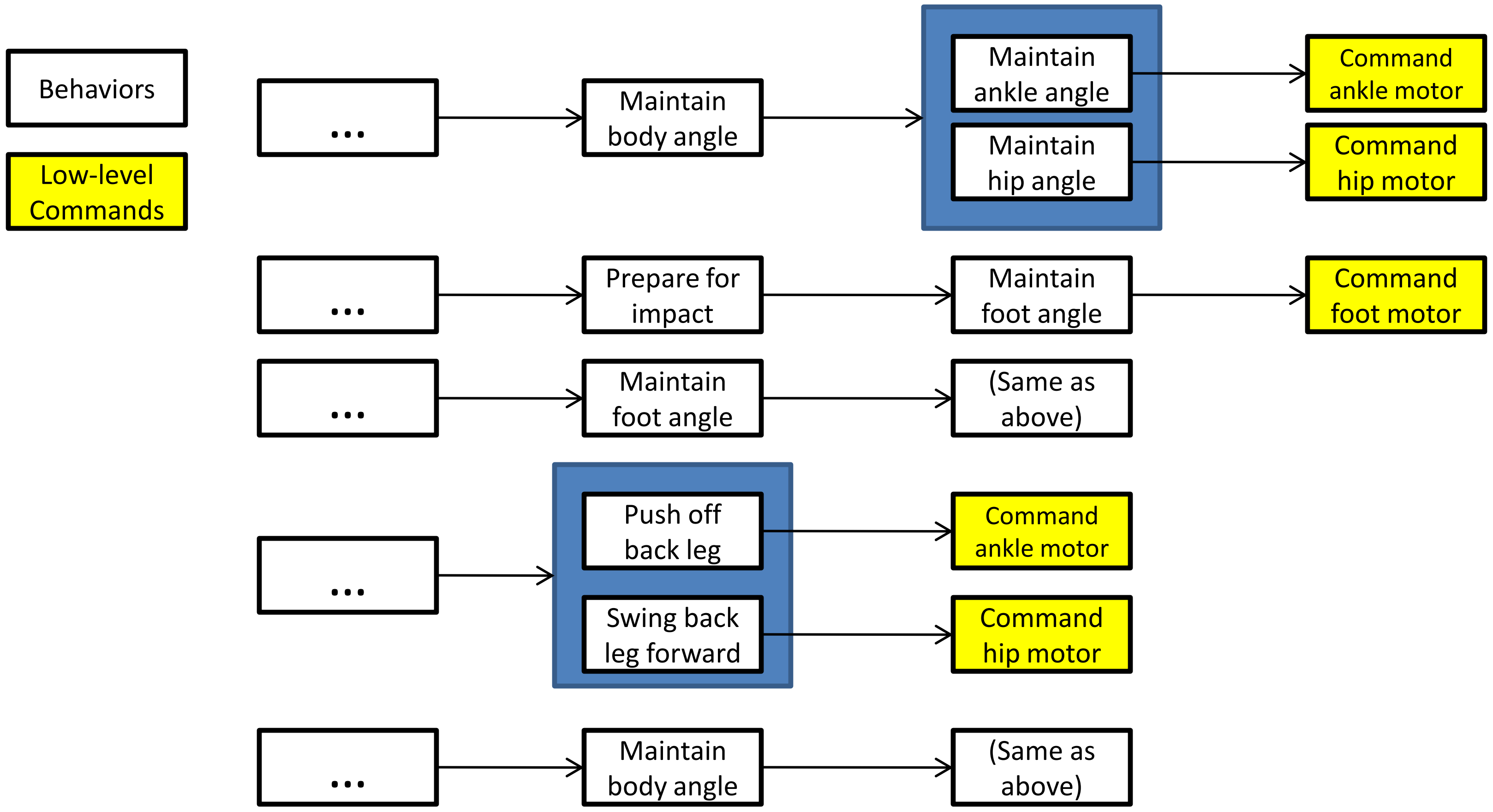
Behavior Trees

- The behavior tree represents the global states and its available actions
- Behaviors have inputs and outputs and may contain control logic
  - Example: “Lean” action takes inputs “direction” and “speed of fall.” It then determines the ankle and hip angles to request based on the speed of the fall.



Behavior Trees

- The leaves of the tree are the actuator commands
- Behaviors determine the parameters to send to the actuators
  - Example: “Maintain body angle” will specify different angles when the global state is “Stationary” versus “Falling”



Benefits

- Less processing
  - Global state machine only checks the key observations that change its global state
- Simplicity
  - High-level state machines are easier to understand and debug
- Code reuse
  - Actions are abstract collections of functions and control logic that can be parameterized for different robots