

A Behavior Architecture for Fast Humanoid Robot Door Traversals*

Duncan Calvert^{a,b}, Luigi Penco^a, Dexton Anderson^{a,b}, Tomasz Bialek^{a,b}, Arghya Chatterjee^{a,b}, Bhavyansh Mishra^{a,b}, Geoffrey Clark^{a,b}, Sylvain Bertrand^a and Robert Griffin^{a,b}

^aFlorida Institute for Human and Machine Cognition, 40 South Alcaniz St., Pensacola, FL, 32502, FL, USA

^bUniversity of West Florida, 11000 University Pkwy, Pensacola, FL, 32514, FL, USA

ARTICLE INFO

Keywords:
humanoid
robot
behavior
software architecture
whole body control
door traversal

ABSTRACT

Towards the role of humanoid robots as squad mates in urban operations and other domains, we identified doors as a major area lacking capability development. In this paper, we focus on the ability of humanoid robots to navigate and deal with doors. Human-sized doors are ubiquitous in many environment domains and the humanoid form factor is uniquely suited to operate and traverse them. We present an architecture which incorporates GPU accelerated perception and a tree based interactive behavior coordination system with a whole body motion and walking controller. Our system is capable of performing door traversals on a variety of door types. It supports rapid authoring of behaviors for unseen door types and techniques to achieve re-usability of those authored behaviors. The behaviors are modelled using trees and feature logical reactivity and action sequences that can be executed with layered concurrency to increase speed. Primitive actions are built on top of our existing whole body controller which supports manipulation while walking. We include a perception system using both neural networks and classical computer vision for door mechanism detection outside of the lab environment. We present operator-robot interdependence analysis charts to explore how human cognition is combined with artificial intelligence to produce complex robot behavior. Finally, we present and discuss real robot performances of fast door traversals on our Nadia humanoid robot. Videos online at <https://www.youtube.com/playlist?list=PLXuyT8w3JVgMPaB5nWNRNHTqzRK8i68dy>.

1. Introduction

The humanoid form has uniquely diverse mobility and manipulation capabilities that drive its suitability as the embodiment of a general purpose robot. This has lead to the pursuit of building humanoid robots to perform useful tasks in spaces designed for humans. One component that is lacking in the literature is how robots can quickly and effectively perform useful whole-body loco-manipulation tasks automatically. There are many challenges in building a system for generating and executing fast loco-manipulation behaviors. Part of the challenge is being able to create behaviors for a wide variety of tasks and to do so in a short timeline as needed. We present an authoring system which contains many features that when put together, provide sufficient capability in this regard. Another aspect of the challenge is in getting the behaviors working on a real robot. We also present an integrated system for testing and iterating on the execution of these behaviors on the real robot.

This topic of behavioral system integration is important to the understanding of how to build useful humanoid robotics systems. There are many aspects of design and some components, if overlooked or misunderstood, could have a negative impact on performance or long term results. As society builds more humanoid robots for military and

industrial purposes, this research could help in grounding system level design principles. Furthermore, we believe this topic of research does not lose its relevance when modern AI approaches are taken, as we think many of the core functionalities and principles explored in this paper are fundamental to humanoid robot loco-manipulation.

A key element to this is *behavior generation* – assembling motor skills to compose tasks and assembling tasks to compose behaviors. In this work we utilize an operator to author behaviors as a tree structure of nodes with general capabilities. This paper explores techniques that build upon known, useful principles in the literature in an effort to nudge the state of the art of behavior authoring on humanoid robots forward. We place a core focus on speeding up the authoring process of behaviors by supporting online behavior generation and modification. By providing a framework for building and modifying behavior trees and action sequences at runtime, most behavioral changes can be done online and quickly enacted on the robot. The increase in authoring speed has allowed us to create more than twenty varieties of working loco-manipulation behaviors on the real robot in the span of one year. The behaviors are primarily different types of door traversals but debris clearing behaviors such as moving trash cans and couches are also included.

Our behavior coordination architecture is designed to create fast, resilient, and reusable loco-manipulation behaviors for humanoid robots as part of a larger effort towards robot that do useful and economical work in the real world. To achieve this, we incorporated state-of-the-art methodologies in building a uniquely complete set of functionalities. The concepts of affordance templates, action primitives,

* This work was funded through ONR Grant 2023-004-01.

✉ dcalvert@ihmc.org (D. Calvert); lpenco@ihmc.org (L. Penco); danderson@ihmc.org (D. Anderson); tbialek@ihmc.org (T. Bialek); a chatterjee@ihmc.org (A. Chatterjee); bmishra@ihmc.org (B. Mishra); gclark@ihmc.org (G. Clark); sbertrand@ihmc.org (S. Bertrand); rgriffin@ihmc.org (R. Griffin)

ORCID(s): 0009-0001-5255-6882 (D. Calvert); 0000-0002-2938-2546 (L. Penco); 0009-0004-1402-0536 (D. Anderson); 0009-0008-1411-2376 (T. Bialek)

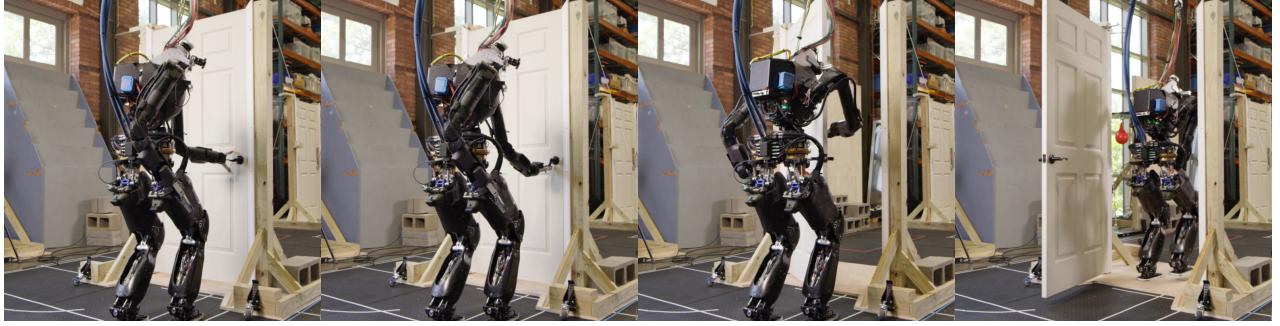


Figure 1: The Nadia humanoid robot performing a right pull lever handle door traversal using cycloidal drive forearms and Sake grippers. Corresponding video: 2024NadiaRightPullLeverHandleCycloidLowRes.mp4

sequential composition, concurrent action layering, behavior trees, and human-robot interdependence were carefully included into the presented novel implementation.

This is currently no body of research available that explores door traversals with bipedal humanoid robots. There are many publications which explore door traversals on wheeled base robots with arms, such as [14], [4] [36], and [21]. We believe this is a critical area of study, as the humanoid form factor is especially well-suited for navigating complex environments encountered in disaster response, space exploration, and urban settings. The advantages of the humanoid structure over wheeled and other mobile platforms are explored further in [18]. Additionally, in this work, we find there are important differences in the planning required for wheeled base and legged robots.

To build modular and adaptable behaviors, we structured them as a composition of low-level actions. Many of these low-level actions are derived from the Affordance Template (AT) framework[10, 11]. ATs are generally a sequence of primitive tasks that are parameterized to a type of higher level task, referred to as an affordance. For example, an affordance template for opening a drawer could be a set of grasp approach, grasp, pull drawer, and let go actions that are available to execute any time the robot sees a certain type of drawer. It is a way of defining general action plans for ahead of time for high level tasks the robot is expected to encounter. Additionally, the AT framework provides an integrated environment for authoring templates and provides a general definition language for robot-agnostic manipulation.

The AT framework is also naturally extensible. In recent years, advanced planners such as for stance and grasp generation, navigation, and motion planning have been integrated[12]. There is also a growing area termed “affordance primitives”[29, 28] (APs) which are a way to model actions with constrained movement, such as turning a valve or closing a drawer. A special AP called a screw primitive has been used to define impedance controlled manipulation of valves and drawers.

ATs have been used on NASA’s Valkyrie humanoid robot to perform a car door opening, retrieve an improvised explosive device (IED) bag, drop the IED bag off in a total containment vessel (TCV) tray, and push the TCV tray in and push its button[20]. The operator could load, edit, and

execute ATs on the fly which provided a human-in-the-loop environment.

To coordinate the ATs, and more in general the low-level actions, we employ Behavior Trees[5, 13] (BTs), which are a popular state-of-the-art model for representing and orchestrating complex behavior. BTs define logical operator nodes such as sequence, fallback, and parallel which work to control the execution flow of a behavior. In BTs, actions are performed by the leaf nodes which command the robot and gather environmental data. Finally, BTs provide reactivity through a ticking system in which each tick starts at the root node. This is in contrast to state machines in which each tick starts from the previous state. By continuously re-evaluating from the top down, from tick to tick there are pathways to ending up in very different parts of the tree without explicit connections between those parts.

Recently, BTs have been used to autonomously navigate cluttered terrain with a centaur wheeled-legged robot[6]. It demonstrates successful navigation on a real hardware system with legs. The behavior tree in this work manages a two-level planner for coordinating intricate locomotion over a debris field of cinder blocks. It includes reactivity to contacts and failures and decides on the next appropriate action to achieve a level of resiliency. However, we are not aware of any prior work that illustrates an application of BTs to loco-manipulation behaviors on bipedal humanoid robots.

Behavior trees and low-level actions are integrated into an interface grounded in Coactive Design (CD)[17, 19] principles. CD is a methodology which gives structure to the design and engineering of systems from the perspective of the interdependence of humans and machines. CD is an iterative process that is comprised of three main steps: identifying, documenting, and iterating on interdependence relationships. Interdependence analysis (IA) charts are used to document the interdependence relationships, which illustrate the connections between system components. CD was previously used for the design and development of the operator interface used by IHMC in the 2015 DARPA Robotics Challenge (DRC) Finals. A later analysis details how the methodology led to success in the competition[18].

MIT’s DRC team developed an alternative system called Director[24] which used AT concepts in a framework to integrate robot autonomy components and task execution.

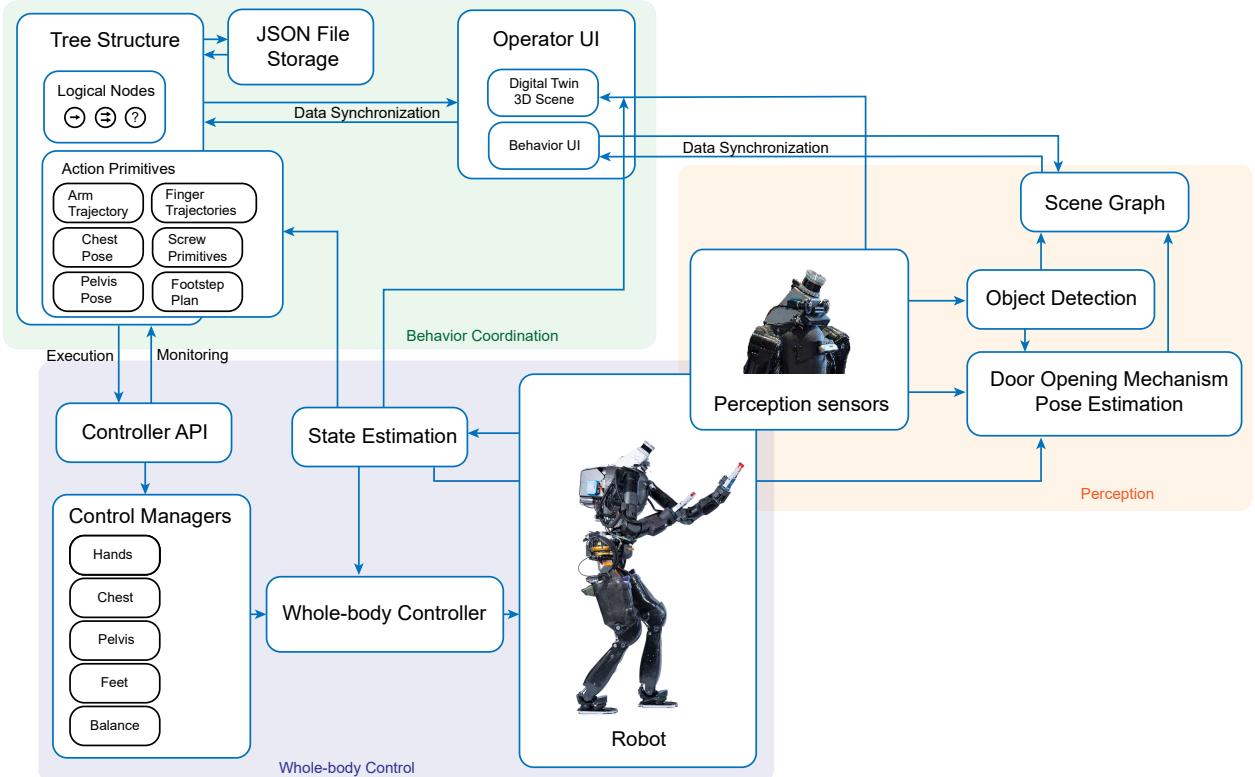


Figure 2: An all inclusive overview of the parts involved in this work.

Director had increased autonomy compared to tools used by other teams in the DRC. It featured advanced planners for manipulation and locomotion, an “operator in the loop” pipeline for executing actions, a 3D scene with interactive widgets, and a way to make custom panels of widgets. A Python program editor was embedded directly in the user interface to write scripts for tasks at the behavior level.

A well-designed interface that supports the robot’s directability and observability is crucial, but equally essential is the capability to accurately detect objects, as they often serve as key points of interaction during tasks. YOLO-based methods have recently become popular for detecting doors[2]. When combined with the planar region extraction of the door panel, the position and orientation of door handles can be obtained. [21] uses YOLACT image segmentation to obtain the door handle position and a RANSAC plane segmentation algorithm to obtain the orientation. [25, 26] present a planar region extraction algorithm which can be used to detect door panels. Together, these detection and segmentation methods provide the foundational perception capabilities needed for effective interaction with doors.

In this paper, we present a system for authoring and operating bipedal loco-manipulation behaviors. To validate our approach, we present and demonstrate a variety of fast door traversal behaviors on real hardware. Our implementation builds on state-of-the-art techniques and includes several novel contributions:

- To enable increasing the speed of humanoid door traversal behaviors, we developed a system for an operator to compose concurrent sequences of locomotion and manipulation action primitives.
- Our integrated behavior framework enables the operator and robot to author reusable, fast, and resilient whole body loco-manipulation behaviors for types of doors that have not yet been encountered.
- To detect door opening hardware well enough to construct and perform door traversal behaviors outside of a lab environment, we developed a novel detection system that combines YOLO-based object detection and segmentation, depth correspondence, and rapid planar region extraction.

2. Architecture Overview

Our system, illustrated in Figure 2, spans three major domains: behavior coordination, perception, and whole body control. We coordinate robot behavior using an interactive tree-based runtime which contains logical nodes and primitive actions. The on-robot runtime is synchronized to an operator user interface containing a digital twin of the robot and environment, direct teleoperation elements, and a behavior management interface. The behavior management interface facilitates managing both the authoring and execution of behaviors. The perception system is a novel

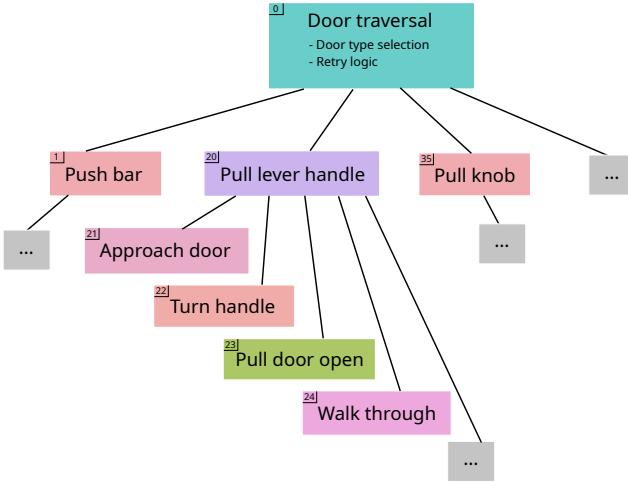


Figure 3: The general structure of a tree for door traversals. A high level node triages control to subtrees for particular door types and takes action in failure scenarios to retry set of actions or change strategy.

assembly of a YOLO neural network, planar region extractor, and heuristics for detecting door opening mechanisms. These system elements interface with a model-based whole-body controller, walking algorithm, and the Nadia hardware platform.

2.1. Tree-Based Runtime with Concurrency

The tree structure is the base of the behavior representation, which is comprised of logical and action primitive nodes. At runtime, logical evaluation starts at the root node and propagates downward as determined by the node implementations. This logic is used to program reactivity and semantic decision making. These nodes have access to perceived object frames so they are able to query knowledge about the environment and make decisions about what to do next. When tasks fail, sub-sequences can be re-executed or alternative sub-trees of behavior can be invoked.

In contrast to BTs, our implementation updates all nodes on each tick and does not use predefined return types ("success", "failure", "running", etc). This supports a more free-form implementation with less formal structure. Instead, ours nodes implement a common interface for storing children, saving and loading with JSON files, executing on the robot side, and rendering user interface elements on the operator side.

Figure 3 illustrates the tree structure used for door traversals. At the root, we place a logical door traversal node which manages the various supported types door traversals. It is able to trigger and monitor the execution of its children and access the scene knowledge graph from the perception system. Reactive and robust logic is then programmed into the door traversal node which enables resiliency through retrying and employing alternative strategies in response to failures.

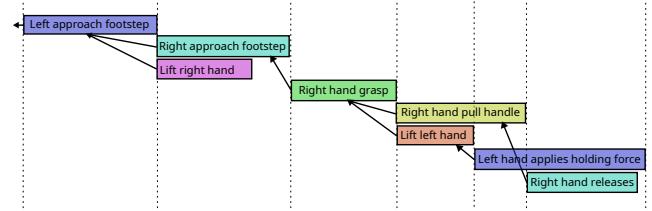


Figure 4: To achieve layered, concurrent execution while maintaining a sequence structure, each node has a pointer to a node to execute after. When an action is next for execution, it can start when the action it's waiting for is complete. The first part of a left pull handle door is illustrated as an example.

To speed up execution and enable more capability, our action executor features layered concurrency. This is accomplished with an "execute after" field on each action primitive node which points to a prior action as a dependency on its completion. This keeps actions starting in order but potentially earlier. We illustrate this in Figure 4 and define Algorithm 1.

Algorithm 1 Autonomous Concurrent Layered Execution

```

Let  $A$  be the set of actions  $\{a_0, a_1, a_2, \dots\}$ 
 $i \leftarrow$  index of the next action to execute
function UPDATE
    while  $i < \text{length}(A)$  do
        if  $\text{SHOULDEXECUTE}(a_i)$  then
             $\text{EXECUTEASYNCHRONOUSLY}(a_i)$ 
             $i \leftarrow i + 1$ 
        end if
    end while
end function
function SHOULDEXECUTE(action)
     $a_{\text{after}} \leftarrow \text{GETEXECUTEAFTER}(action)$ 
    return  $\neg \text{ISExecuting}(a_{\text{after}})$ 
end function

```

2.2. Action Primitives

The primary interface between the behaviors and the controller are the action primitives, which are the leaf nodes in the tree and the pieces that trigger execution and handle monitoring execution. The action primitives include footsteps and arm, hand, chest, and pelvis trajectories. Through the authoring process, this primitives can be assembled into ATs. The behavior author is responsible for assembling sequences of actions the ensure inherent stability when executed.

[3] discusses the guarantees of stability of non-linear dynamic systems when multiple controllers are strung together. It is described and illustrated as cascading Lyupanov funnels as shown in Figure 5 where system states are accepted in boundaries at the top and are controlled to a state that is acceptable by the next controller in the sequence. Towards modelling our system in this way, our primitive actions each implement entry and exit conditions. On-entry conditions

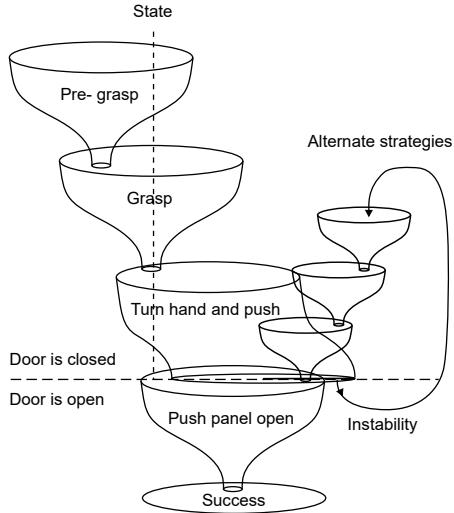


Figure 5: An illustration of how we apply sequential composition. During authoring, actions are carefully placed in an order in which the Cartesian result of the action informs the robot about the state of the environment such that it can automatically decide to proceed or not. When control is insufficient, the state will not fall into the stable region of the next action and a high level node will either retry or employ alternative strategies.

serve to gate action execution with checks that determine the feasibility while on-exit conditions determine success or failure.

Furthermore, these actions are setup in task relative frames to provide re-usability in a particular environmental context. We provide the behavior author with a 3D pose gizmo shown in Figure 6 to allow them to interactively edit the task relative transforms.

To orchestrate walking, we implemented a footstep plan action that represents a sequence of footsteps that get queued to the controller. Frame-relative footsteps can be pre-specified or planned online. Online planning is provided by the IHMC footstep planner[9] by specifying a frame relative goal stance. This planner can be setup to run online, enabling dynamic planning in the environment, or run once during authoring to pre-plan the footsteps. A procedural "turn-walk-turn" planning mode is also available for flat ground scenarios. To adjust the walking speed and reliability, the operator is also able to tune the nominal swing and transfer duration of the steps.

We found that the perceived object frames can have significant enough orientation error that the goal stance steps do not align with the ground well enough for robust walking. To account for this, when online planning is active, we snap the goal stance vertically onto the ground using two frame-relative points. We define these as a point to face and a point to stand at. These can be seen with a white arrow connecting them in Figure 7. The operator may also adjust the relative X-Y position and yaw of the stance steps with the gizmos as shown.

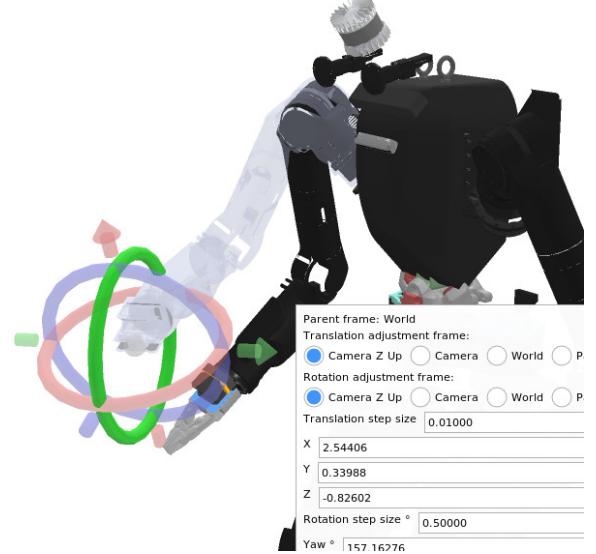


Figure 6: A gizmo for manipulating the hand's 6 DoF pose with a mouse and keyboard. The mouse can be used to drag the rings and arrows to perform constrained adjustments to those axes. The keyboard arrow keys can be used to move the gizmo with respect to the adjustment frames selected in the context menu. Modifier keys Ctrl, Alt, and Shift are used to select rotation axes and adjust the sensitivity. The context menu shown on the right allows the user to change the frame of adjustment, nudge the pose by small amounts, and view and edit the absolute values of the pose. The IK preview of the arm is also shown as a transparent graphic.

An arm trajectory action is used to move the arm and hand to either preset joint angles or an online IK solution for a frame-relative hand pose. Specified for a single arm at a time (left or right), a 3D gizmo is used to adjust the desired hand pose in IK mode as shown in Figure 6. A trajectory time setting is used to adjust the speed of the motion. To adjust the whole body controller's tracking priorities, tunable weights and gains are also provided.

Inspired by [28], we also implement a parameterized helical screw trajectory option, illustrated in Figure 8. This allows the operator to define revolving motions useful for turning handles and swinging door panels. It is defined by a 3D axis (dashed white line), an angle of revolution, and a translation amount to move along the axis. The trajectory is also grasp-invariant, as it generates the trajectory online from the hand's pose just before execution.

Arm trajectories are often combined with chest and/or pelvis pose trajectory actions, which help to increase the reachability workspace. For example, in the pull door behavior, the right arm moves across the robot to the handle on the left side of the door, aided by a chest yawing motion.

We control the hands with an action that tracks finger trajectories and preset hand configurations. The operator can specify the joint angles of the knuckles or choose from a list of configurations such as "open" and "closed". Additionally,

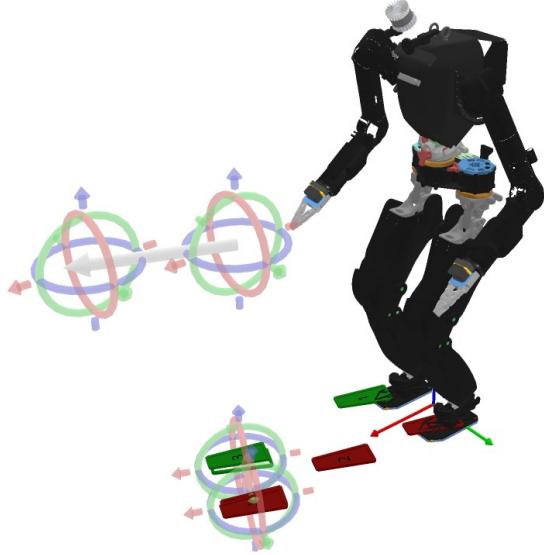


Figure 7: The approach stance AT is authored via a frame-relative point to stand under and point to specify the facing direction. The X-Y position and yaw of the stance footsteps are defined by the operator but are otherwise snapped to the ground.

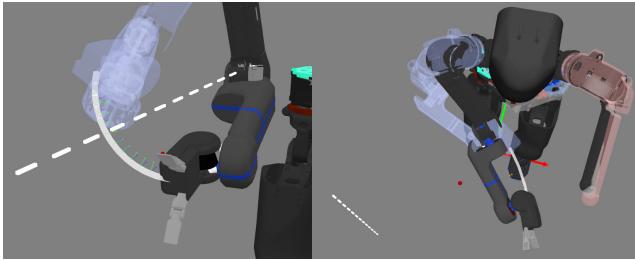


Figure 8: A visual rendering of generated screw primitive trajectories about two common axis configurations which are similar to those used for turning lever handles and opening doors by pulling the panel in a revolving motion.

a maximum torque option is available, which allows to operator to adjust the grip strength.

2.3. Whole Body Control

The behavior coordination layer was built to interface with a whole body and walking controller that is able to execute arbitrary trajectory requests and footsteps. The behavior runtime process sends requests to the controller and monitors execution through statuses sent back from the controller.

We use the whole body momentum-based control framework developed by Koolen et. al.[23]. The general structure and control flow of the whole body controller and walking implementation is illustrated in Figure 9 and Figure 10. It is a model-based torque control scheme centered around a quadratic program (QP) that reconciles motion tasks expressed as constraints on the joint acceleration vector. To manage the robot's interaction with the environment, we

manage additional QP constraints for foot-ground contact and force-limited grasping with the hands. The action primitives described previously make use of motion tasks provided by this controller which include jointspace and spatial (taskspace) acceleration objectives.

We use the dynamic walking behavior developed by IHMC and described in [23]. It is comprised of a state machine with standing, transfer, and swing states where each has associated motion tasks and active contacts that are used to achieve balance. The controller is setup to execute a queue of footstep poses submitted by the user.

To model balance dynamics, it uses instantaneous capture point (ICP)[31]. Foot pose, pelvis height, and pelvis orientation trajectories are generated to each achieve the user desired footsteps while maintaining balance. During transfer an ICP trajectory is also generated, along with a relaxation of stance foot contact constraints to allow the heel to lift. The centroidal moment pivot (CMP) is also be used to exhaust the body's ability to control balance with its momentum. More details can be found in [32].

Foot swings are executed referencing smoothed trajectories which vary by terrain type. On rough terrain, foot swings may be planned to avoid tripping on obstacles. On flat ground, default, conservative trajectories are used that have low swing heights. To recover from external disturbances and instabilities, swings can also be sped up during execution to more quickly place the foot and catch balance.

Lastly, for state estimation, a combination of a kinematic foot-ground contact proprioception for pelvis position and a pelvis-mounted IMU for pelvis orientation is used. When a foot is in contact with the ground, determined by a glitch filtered and delay compensated ankle torque sensor based virtual foot switch, we assume the foot does not slide. With one or both feet on the ground and assumed still, the pelvis position is computed simply via forward kinematics. Consequently, the positional estimate of the robot is susceptible to drift in three main ways: inaccuracy of foot contact detection, foot slipping, and lack of both feet being on the ground.

2.4. Operator Interface

Our operator interface is shown in Figure 11, which allows a human operator to construct and execute behaviors. It contains a digital twin of the robot and the environment rendered in 3D as well as panels of interactable widgets for teleoperation and interacting with behaviors. The operator can execute actions one at a time or autonomously. While the robot is performing, tracking plots, log messages, flashing colors, and other items-of-importance are presented to the operator in real-time.

A unique element of our interface is that it allows behavior creation, modification, and execution online. This feature speeds up the development process as there is no re-deploy step to run a new behavior. This means that changes to any part of behaviors can be tried immediately and iterated on quickly on the real robot. This “on the fly” authoring

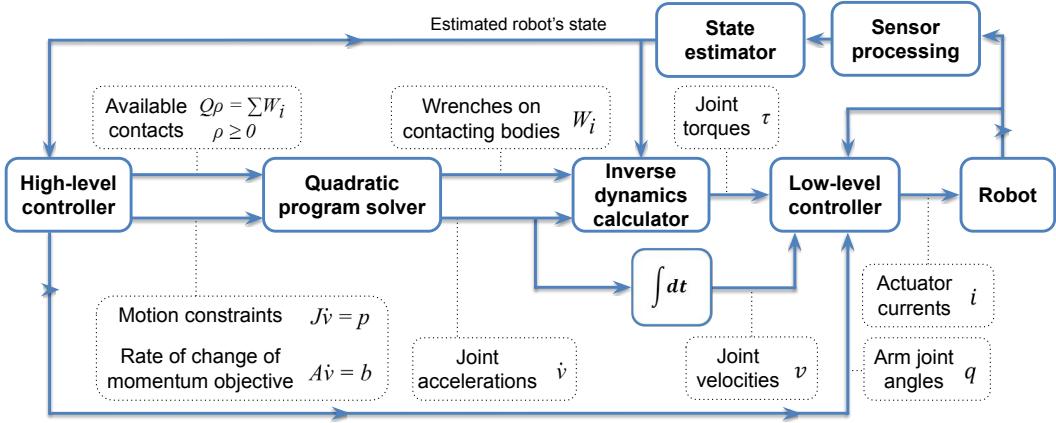


Figure 9: IHMC overall whole-body control framework.

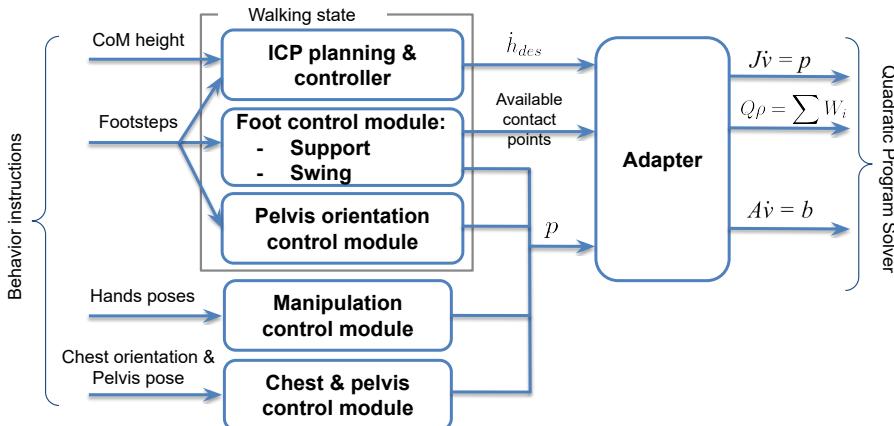


Figure 10: IHMC humanoid walking controller, highlighting several different control modules that regulate mobility and manipulation. Outputs of these control modules become input to the quadratic program solver in the whole-body control framework.

also supports the operator-robot team to accomplish and automate never before seen tasks as they are encountered.

To preview what the robot will do next, we render a transparent robot and white trajectory graphics, which go through the motions of upcoming actions. This helps the operator understand the resultant motions of primitive actions, the spatial occupancy of the robot, and the task-relative alignment of the goal pose. Widgets are available to scrub through and playback the preview.

As shown in Figure 12, action execution tracking is plotted in real-time to keep the operator informed about progress. We plot the Cartesian distance to the goal as one value rather than the separate components X, Y, and Z to help the operator focus on the overall tracking error. A progress bar shows the remaining nominal execution time. Red and green are used to indicate error and good tracking, respectively to further assist the operator. A log area is also available that shows printed, colored messages from the robot's execution process.

Behaviors are stored as hierarchical JSON files which can be selected and loaded and saved. Once the root node is created or loaded, the operator is presented with the UI

shown in Figure 11. Context menus allow the operator to add, remove, and reorganize nodes in the tree. The tree and all of the nodes' properties and execution data is automatically and continuously synchronized with the robot's working copy.

There are two main modes for executing behaviors. To test and iterate on behavior parts, a manual step-by-step mode is used by clicking a button to execute the next action. The operator has an option whether to allow action concurrency in this mode. The other main mode is the autonomous execution mode, where the robot will continue to execute actions as decided by the behavioral logic.

3. Application to Door Traversals

In this work, we consider door traversals where the door is closed at the beginning. This is partly enabled by a perception system which can detect door opening mechanisms. Door behaviors are generally composed of a sequence of approach, grasping and unlatching the door, swinging open the panel, and walking through.

Fast Humanoid Robot Door Traversals

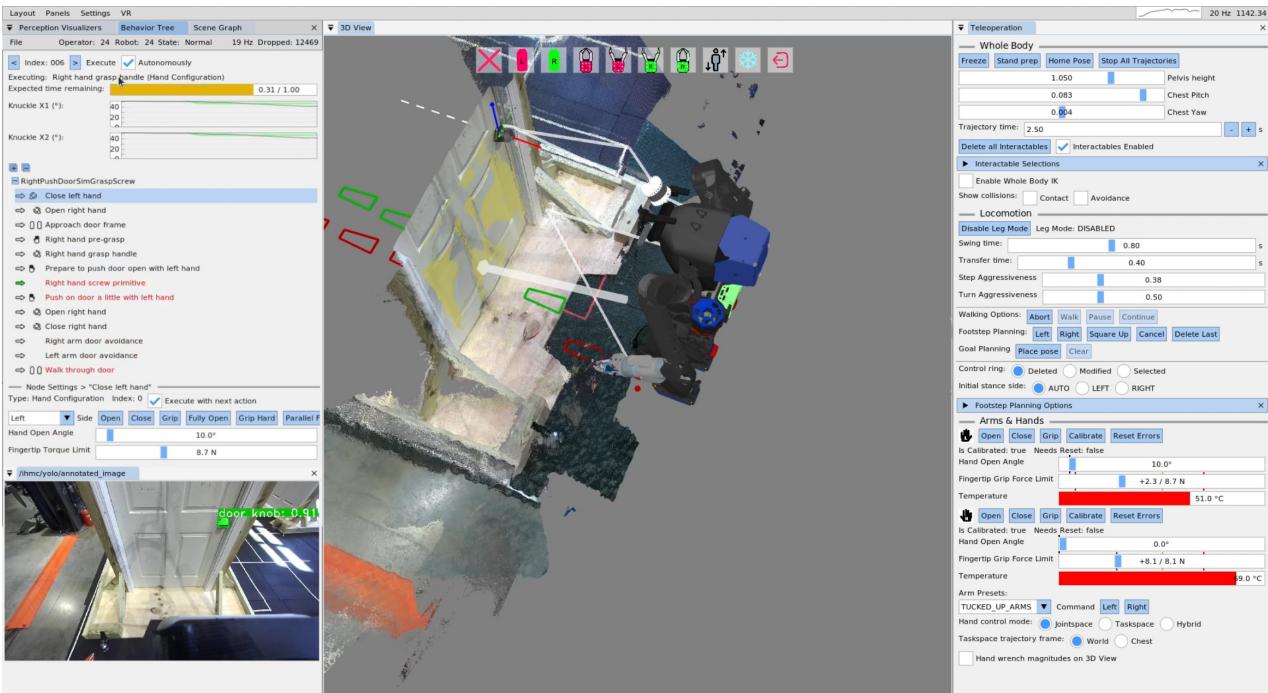


Figure 11: The operator interface used to author behaviors. On the left, the behavior tree editor allows to edit and operate the behavior tree. In the center, the 3D scene shows affordance template elements projected into a scene with the live robot state and perception data. On the right, the teleoperation panel provides controls to directly operate and monitor the robot.



Figure 12: Live tracking plots in the UI help the operator monitor progress, identify failures, and design improvements. Top: Good tracking of a arm-hand trajectory. Bottom: Good tracking of a footstep plan. Discrete jumps represent steps.

Many features of our behavior architecture have been developed through a Coactive Design (CD) process focusing on the human and robot team. The Interdependence Analysis (IA) charts in Figure 13 show connections between components involved in authoring and operating the door traversal

behaviors. The arrows pointing left towards the operator represent observability and predictability while the arrows pointing right toward the robot represent directability. Red cells in the robot column represent capabilities that, if added, would improve reliability or efficiency.

The door approach task is defined by stance footsteps with respect to the pose of the door opening mechanism, which can be detected by the perception system or manually specified by the operator. Online footstep planning is used to accommodate a wide range of initial robot stances. For a push door, a squared up or staggered stance can be used, but for a pull door, the feet must be out of the way of the panel as it swings toward the robot. A good general stance is off to the side of the door that the opening mechanism is on, with staggered feet.

The process of tuning the grasp of the door opening mechanisms is one of the most sensitive components, since many handle and hand combinations require a very precise alignment. The operator typically uses an iterative process of manually tuning and triggering the grasp action until the result is reliable. Grip force, trajectory duration, and hand orientation are primary parameters for this part.

Once the handle grasp action is ready, for a turnable knob or lever handle, the operator will use two screw primitive actions to unlatch and open the door. To simplify the process, the screw primitives are designed to be grasp invariant. Tuning the screw primitive actions primarily involves aligning the screw axes, shown as white dotted lines in Figure 8, to the axes of rotation of the handle and panel. Once aligned, the

Fast Humanoid Robot Door Traversals

Authoring Door Behavior

Phase	Operator	Interface	Robot
Identify affordance / Generate whole body strategy	Identify door type	First-person view video and colored point cloud	Sense stereo color and depth
	Conceptualize door traversal method	Interactable detected opening mechanism in 3D scene	Detect and track door opening mechanism pose
	Create new behavior		
Generate approach stance	Plan for task reachability	Interactable approach footstep plan with stance placement gizmo and planning parameter tuner	Generate approach stance
	Plan stance steps that avoid collisions throughout task		Execute the approach footstep plan
	Test and iterate on task component	Operational control and status widgets	
Generate grasping strategy / Open the door	Plan grasp, unlatching motion, and panel swing open	Interactable body, arm, hand, finger trajectories with previewing and tunable parameters	Generate grasp configuration
	Test and iterate on task component	Plot execution data and display colorful statuses	Turn handle with force feedback until unlatched
		Operational control widgets	Execute door opening trajectories
Hold the door open / Walk through the frame	Plan collision-free footsteps	Interactable arm trajectories and footstep plan with previewing	Detect and track door panel and frame
	Plan arm-hand motions to hold door	Operational control widgets	Plan collision-free steps through door
	Test and iterate on task component	Display success or failure status	Plan arm motions to hold door open
			Execute door traversal footsteps and arm motions

Operating Door Behavior

Phase	Operator	Interface	Robot
Select and trigger behavior	Place the robot to within visible range of door	Indicate door type	Detect door opening mechanism
	Select behavior and trigger execution	Colored point cloud of environment and detected door opening mechanism	Door traversal triggered by navigation algorithm
Monitor execution	Identify when behavior needs improvement and go back to authoring process	Operational control widgets	Execute behavior
Evaluate task completion	Evaluate success and performance quality	Plot execution data and display colorful statuses	Report success or failure
		Display success or failure status	

Figure 13: IA charts for authoring and operating pull door behaviors.

Color Key for Team Member Role Alternative Capability Assessment	
Operator	Robot
I can do it all	My assistance could improve efficiency
I can do it all but my reliability is < 100%	My assistance could improve reliability
I can contribute but need assistance	My assistance is required
I cannot do it	I cannot provide assistance
Not applicable / Not significant	Not applicable / Not significant

Figure 14: IA chart color key for team member role alternative capability assessment.

angle of rotation is determined and the motions are iterated on until satisfactory.

The next and final phase of a door traversal is to walk through while using the arms to keep the panel out of the robot's way. The first element of this is to persistently keep an arm in the way of the door closing again to prevent it from re-latching. The second element is to walk through, with an arm or shoulder designated as a contact surface which keeps the door panel out of the robot's way.

Our robot's shoulders were 85% as wide as the door frame opening which meant our normal walking footstep width resulted in collisions between the robot's shoulders and the door frame, causing the robot to fall. As illustrated and modelled in [8], the center of mass (CoM) of the robot, located near the torso, will sway side to side during walking as an artifact of balancing. As shown in Figure 15, both taking narrower steps and having the steps straddle the door threshold help to keep the shoulders centered with respect to

the door frame and avoids collisions. To do this, we utilized the manual step placement feature of the footstep plan action primitive.

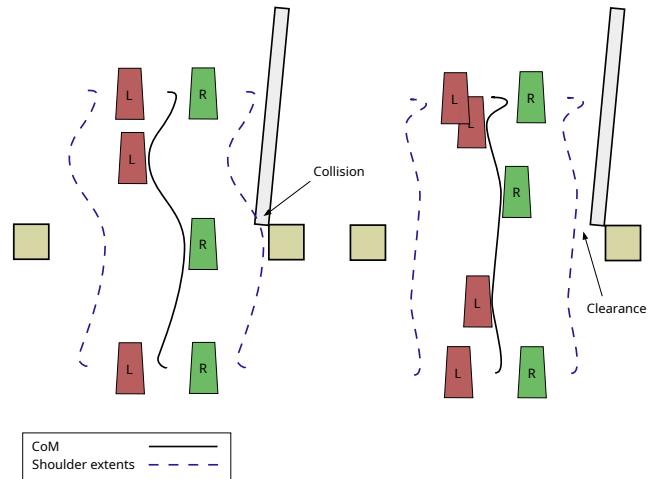


Figure 15: When walking through the door, narrower steps result in the robot's body (CoM) swaying less and resulting in collisions with the door frame. Additionally, straddling the door threshold with the the footstep keeps the body centered in the door frame during the transition.

4. Door Handle Detection

To enable this presented behaviors, we developed a novel detection and tracking system for door opening mechanisms. When tracked, the poses of detected door opening mechanisms are provided to the behavior runtime for use in authoring and execution of behaviors. We developed models to detect and track the four different opening mechanisms shown in Figure 16.

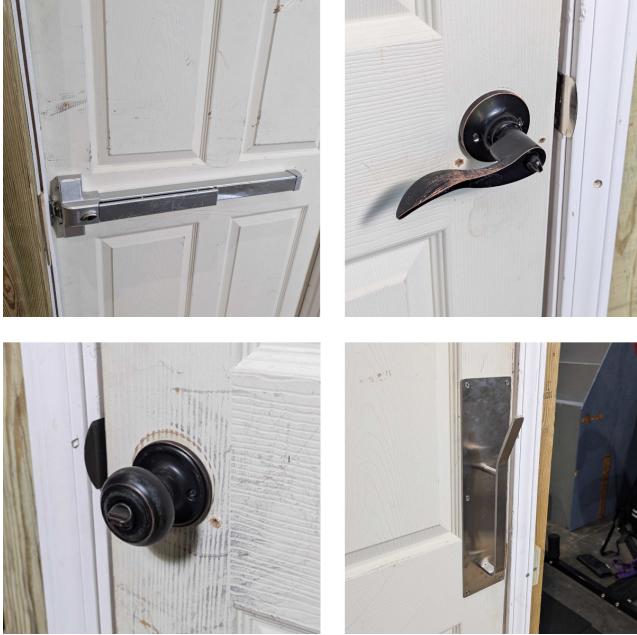


Figure 16: The four types of door opening mechanisms used: a push bar (top left), a lever handle (top right), a knob (bottom left), and a hands-free pull handle (bottom right).

To detect and track the pose of door opening mechanisms, we use an 8 step process illustrated in Figure 17. We use a ZED 2i[35] facing forward and downward at 43° to readily view the area in which door manipulation occurs. The color data from the sensor is fed through YOLO to obtain a segmentation of the mechanism and a planar region extractor is used to obtain the plane of the door panel. The centroid of the depth points on the mechanism is used for the position and the normal of the plane of the door panel is used for the orientation.

In this work we re-trained YOLOv8[33] with data for our specific handle types. The generate synthetic data for training, we scanned objects using Polycam[30] and rendered them in photo-realistic scenes using Blender[7]. In each scene we also placed objects to serve as distractors, including primitive shapes and selections from Google Scanned Models. An example render is shown in Figure 18. Thousands of images are rendered and associated with the corresponding ground truth object pose, binary segmentation image, occlusion-aware bounding box, and camera intrinsics.

To get a high quality result, we also included annotated real world data. This data was gathered with the ZED 2 by

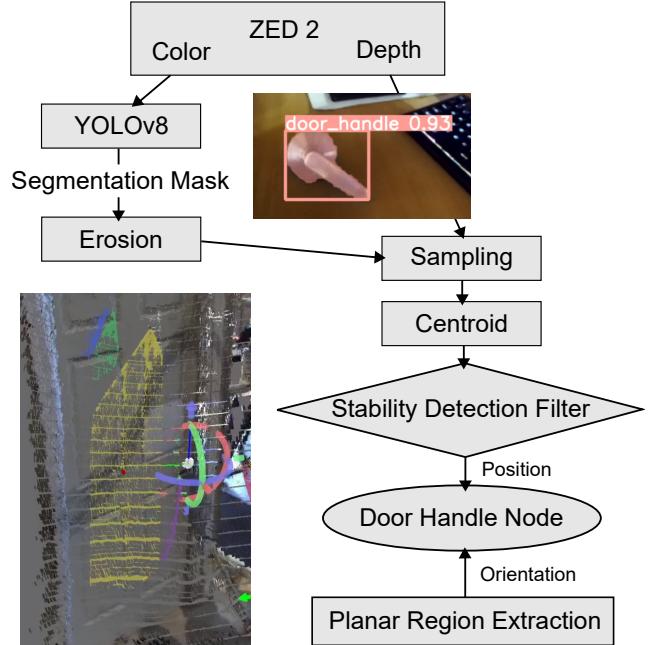


Figure 17: Our detection process uses ZED 2 color and depth data, YOLO, centroid calculation, stable detection filtering, and fusion with planar regions to estimate the pose of the door mechanism.



Figure 18: An example rendering of a door lever handle in Blender with randomized background and distractor placement.

capturing images of the door and handles with varied door positions, openness, lighting conditions, and background. We obtained the segmentation labels by manually selecting the region of interest and running Segment Anything[22] on that area. To improve training, for 15% of the samples, we used images that did not contain a door or a handle. We found that a combination of 1000 real and 1000 synthetic labeled samples were enough to train YOLO for sufficient performance.

To calculate the position of the door opening mechanisms, the resulting YOLO segmentation is eroded and combined with the corresponding depth points from the sensor. The points are sampled down to a smaller number

and a centroid is taken. The centroid is then alpha-filtered to reduce noise and improved the stability of the result.

To obtain the orientation of the mechanism, the rapid planar region extractor from [25] is run with the depth data from the ZED 2i sensor. For a mechanism detected by YOLO, we take the positional calculation and search for a planar region that is nearby, has a normal perpendicular to gravity, and is sufficiently large to plausibly be the door panel. If such a planar region is found, the orientation is set as the mechanism's pose. To provide the behaviors with a stable and reliable detection, we require that both YOLO detects the mechanism and a valid planar region association is made for at least 60% of the sensor frames within a one second window.

5. Results

The videos demonstrating our results are available here. We performed six variations of door traversals by varying push and pull with left and right handed hinges and with knob, push bar, and lever handle opening mechanisms.

We performed these experiments on the Nadia humanoid robot developed by IHMC with Boardwalk Robotics. It features 6 degree of freedom (DoF) legs, a 3 DoF spine, 7 DoF arms, and 1 DoF grippers. The ankle, knee, hip pitch and roll, and spine pitch and roll joints are actuated by Moog Integrated Smart Actuators (ISAs)[27]. The hip and spine yaw joints are actuated by electric motors with planetary drives. The arms are powered by cycloidal drive actuators, described in [1], which provide high torques, near-zero backlash, and robustness to heavy impacts. SAKE EZ Grippers[34] are used for the hands. The robot features a carbon fiber exoskeleton for the thighs, shins, and torso to save on weight and enable contact with the environment.

Because our robot is tethered to a hydraulic pump and cannot sustain falls, we built door mock ups without the top part of the frames. The lab doors are otherwise designed using real door hardware commonly found in residential and commercial buildings. Additionally, spring closers are mounted and were used for some of the experiments.

Video key frames of a right pull lever door traversal are shown in Figure 1. The corresponding tree is illustrated in Figure 19. The sequence is composed of three main phases: approach, unlatch and open door, and walk through. Wait actions are added to allow the perception system to get a more accurate pose of the door handle and increase reliability.

Our door traversal behaviors execute on the timescale of tens of seconds. The distance over time of four representative types of door traversal behaviors on the the real robot are plotted in Figure 20. Push door behaviors are the quickest, executing in less than 20 seconds, because the robot can keep making forward progress. Pull door behaviors take about 10 seconds longer because the robot must stop further back, stay clear of the door panel, and open it fully before proceeding.

During traversals with spring loaded door closers, the robot can experience significant disturbances. To show the degree of balance disturbance, an overhead plot of the center

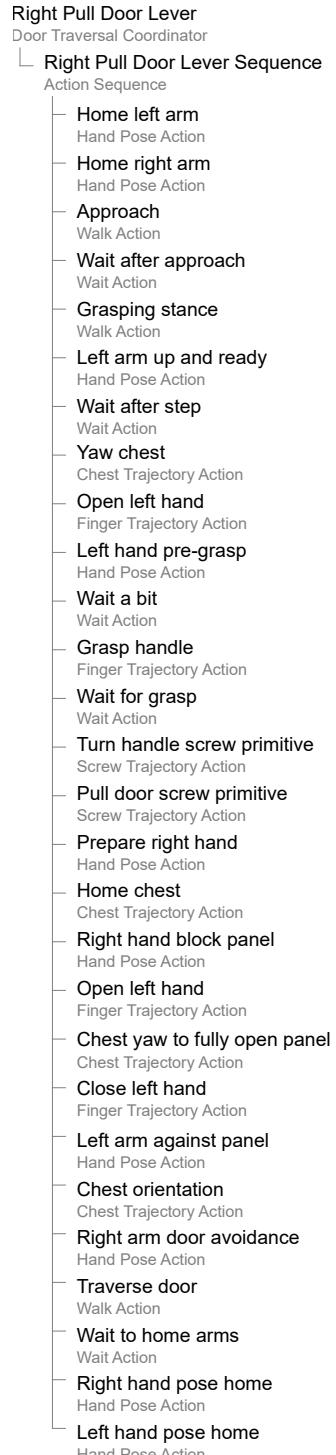


Figure 19: The tree for the right pull lever door traversal behavior. Note the actions are listed sequentially but scheduled dynamically and concurrently as shown in Figure 4.

of mass and instantaneous capture point trajectories for a right pull handle door traversal is shown in Figure 21.

We also ran three door traversals consecutively in a single composite autonomous behavior. Video key frames of this three-in-a-row door traversal run are shown in Figure 22.

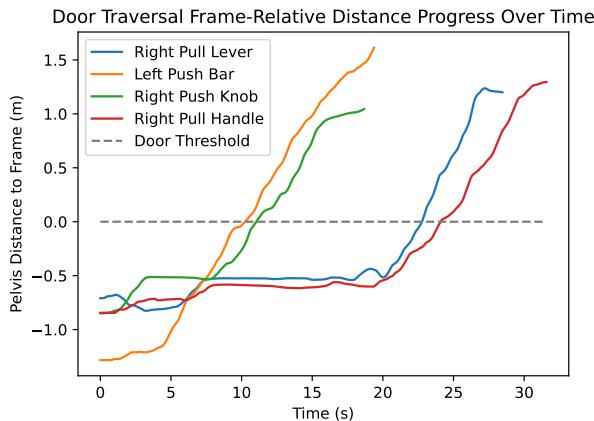


Figure 20: Plots showing behavior speed as distance progress through the door frame. Four door behavior runs on real robot of various types are shown.

To demonstrate the generality of the approach, we included a debris clearing behavior as part of a right pull handle door traversal. In Figure 23, the robot can be seen executing this behavior which pushes a recycling bin out the way of the door. This experiment included a building exploration node that identified that the recycling bin was blocking the door, executed the behavior to move it out of the way, and then executed the door traversal behavior.

6. Discussion and Future Work

Our work has enabled a bipedal humanoid robot to traverse doors in 20-30 seconds, which is many times faster than the prior methods used by the IHMC Robotics lab, including in the DARPA Robotics Challenge Trials in which IHMC Robotics won the “Best in Door” award. We think a variety of factors lead to this dramatic increase in speed. First, the Nadia robot hardware features higher maximum joint velocities than DRC Atlas and NASA Valkyrie, enabling it to walk and move faster. Second, the 7 DoF electric arms with cycloidal drives provide good compliance, high torques, and high speeds, enabling manipulation motions to be fast and robust to unplanned contacts. Third, the concurrent action implementation discussed in this paper enables the robot to execute layered actions as soon as dependent actions complete, eliminating pauses and doubling speed. Fourth, the coactive UI design has allowed for rapid iteration of these behaviors, enabling us to find and implement faster strategies more quickly. Last, advances in GPU accelerated perception have allowed the robot to identify door features both directly and quickly, removing even more long pauses.

Despite great results in fast door traversals and a framework for generating general loco-manipulation behaviors, there is still a considerable amount of improvements that

could be made. These improvements focus mainly on increasing executing speed, decreasing dependence on a human operator, and increasing the capability to perform both dexterous and high powered manipulations.

We are still far from achieving human speed and performance, which would be necessary for a robot to stay with a group of humans in urban operations. Major slowdowns in this work are unnecessary footsteps, pauses, and maximum walking speed. One way to mitigate this would be to eliminate steps that are not necessary and keep the robot walking by blending the footstep plans between actions. The robot should only take steps that are purposeful and of utility.

In this work, we defined trajectories primarily in position space, but we believe moving to force and impact based action definitions will simplify the action definition, increase disturbance robustness, and provide another way to perceive the environment, through proprioception, by gathering reaction force data. Doors, being constrained by revolute joints, would be a fitting candidate for such techniques. Work has been done in this area in the form of impedance controlled actions [15][28].

We also relied heavily on the ingenuity of a human author and operator. One way to mitigate that dependency is to implement more advanced planners, such as stance and grasp generation and collision free motion planning as in [4]. [16] uses a graph based planner combined with an IK trajectory solver for navigating door traversals for wheeled base robots with an arm. [37, 21] use reinforcement learning to train a quadrupedal robot with an arm to traverse doors.

Lastly, we are unable to detect the door frame and thus must assume the door is closed when it’s first seen. A detection algorithm for door frames would enable traversing doors in any initial condition. We speculate that some additional planning heuristics would also be required for doors that are initially ajar.

7. Conclusion

We have presented theory, designs, an implementation, and real world results for a humanoid robot loco-manipulation behavior architecture. A key defining characteristic of our work is the speed of the robot in accomplishing tasks. We think our system is the fastest presented such system in the literature. Ultimately, achieving human speeds and faster in building exploration and other loco-manipulation tasks are key for humanoid robots to become useful.

7.1. Acknowledgements

We would like to thank Jerry Pratt, Stephen Hart, Matt Johnson, William Howell, Thomas McKenna, and the IHMC Robotics team, without whom this work would not have been possible.

7.2. Source Code and Media

Our implementation and the associated modules discussed in this paper can be found on our GitHub at <https://github.com>:

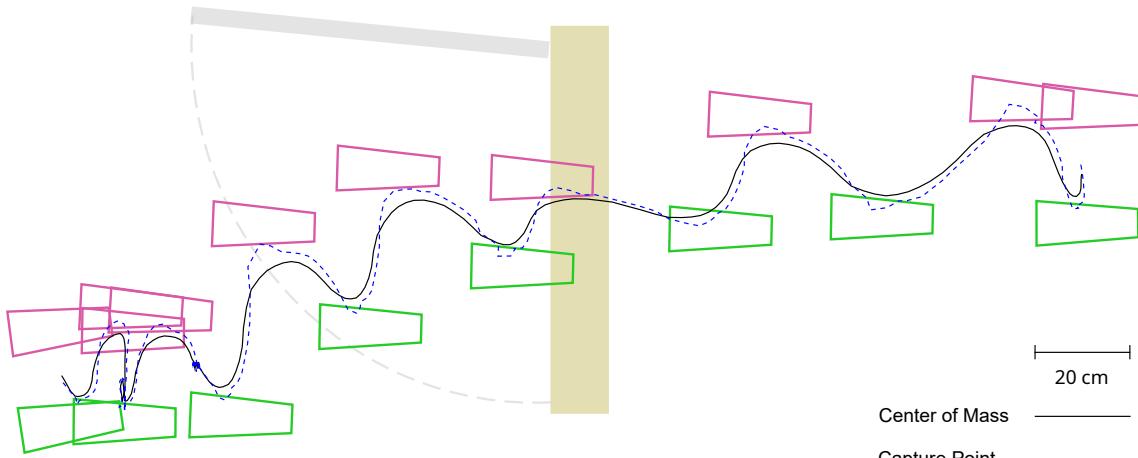


Figure 21: An overhead plot of a right pull handle door traversal. The footsteps and center of mass and instantaneous capture point trajectories are plotted to show balance performance.

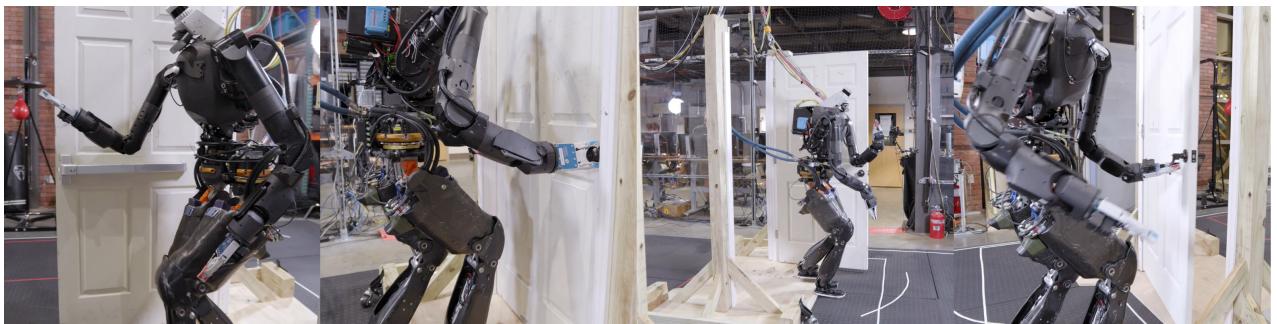


Figure 22: Three door traversals in a row with continuous autonomy. This illustrates the composability of higher level building exploration behaviors. The door traversal types are push bar door right hinge side, right push knob, and right pull lever handle. Corresponding video: 202407NadiaDoorTraversalsPushBarPushKnobPullHandleFallLowRes.mp4



Figure 23: A behavior that moves a recycling bin that is blocking access to the doorway. Corresponding video: 202407NadiaClearTrashCanLowRes.mp4

//github.com/ihmcrobotics. The accompanying videos can be found here.

8. Bibliography

References

- [1] Bertrand, S., Penco, L., Anderson, D., Calvert, D., Roy, V., McCrory, S., Mohammed, K., Sanchez, S., Griffith, W., Morfey, S., Maslyczyk, A., Mohan, A., Castello, C., Ma, B., Suryavanshi, K., Dills, P., Pratt, J., Ragusila, V., Shrewsbury, B., Griffin, R., 2024. High-speed and impact resilient teleoperation of humanoid robots. URL: <https://arxiv.org/abs/2409.04639>, arXiv:2409.04639.
- [2] Bhatti, A.H., Zafar, A.U., Ahmed, M., 2023. Door detection and distance estimation in habitat simulator for 3d environments, in: 2023 3rd International Conference on Digital Futures and Transformative Technologies (ICoDT2), pp. 1–8. doi:10.1109/ICoDT259378.2023.10325708.
- [3] Burridge, R.R., Rizzi, A.A., Koditschek, D.E., 1999. Sequential composition of dynamically dexterous robot behaviors. The International Journal of Robotics Research 18, 534–555. URL: <https://doi.org/10.1177/02783649922066385>, doi:10.1177/02783649922066385, arXiv:<https://doi.org/10.1177/02783649922066385>.
- [4] Chitta, S., Cohen, B., Likhachev, M., 2010. Planning for autonomous door opening with a mobile manipulator, in: 2010 IEEE International Conference on Robotics and Automation, pp. 1799–1806. doi:10.1109/ROBOT.2010.5509475.
- [5] Colledanchise, M., "Ogren, P., 2018. Behavior Trees in Robotics and AI: An Introduction. CRC Press, Taylor and Francis Group. doi:10.1201/9780429489105.

- [6] De Luca, A., Muratore, L., Tsagarakis, N.G., 2023. Autonomous navigation with online replanning and recovery behaviors for wheeled-legged robots using behavior trees. *IEEE Robotics and Automation Letters* 8, 6803–6810. doi:[10.1109/LRA.2023.3313052](https://doi.org/10.1109/LRA.2023.3313052).
- [7] Denninger, M., Winkelbauer, D., Sundermeyer, M., Boerdijk, W., Knauer, M., Strobl, K.H., Humt, M., Triebel, R., 2023. Blenderproc2: A procedural pipeline for photorealistic rendering. *Journal of Open Source Software* 8, 4901. URL: <https://doi.org/10.21105/joss.04901>, doi:[10.21105/joss.04901](https://doi.org/10.21105/joss.04901).
- [8] Englsberger, J., Ott, C., Albu-Schäffer, A., 2015. Three-dimensional bipedal walking control based on divergent component of motion. *IEEE Transactions on Robotics* 31, 355–368. doi:[10.1109/TRO.2015.2405592](https://doi.org/10.1109/TRO.2015.2405592).
- [9] Griffin, R., Wiedebach, G., McCrary, S., Bertrand, S., Lee, I., Pratt, J., 2019. Footstep planning for autonomous walking over rough terrain, arXiv. URL: <https://arxiv.org/abs/1907.08673>.
- [10] Hart, S., Dinh, P., Hambuchen, K., 2014. Affordance templates for shared robot control.
- [11] Hart, S., Dinh, P., Hambuchen, K., 2015. The affordance template ros package for robot task programming, in: 2015 IEEE International Conference on Robotics and Automation (ICRA), pp. 6227–6234. doi:[10.1109/ICRA.2015.7140073](https://doi.org/10.1109/ICRA.2015.7140073).
- [12] Hart, S., Quispe, A.H., Lanigan, M.W., Gee, S., 2022. Generalized affordance templates for mobile manipulation, in: 2022 International Conference on Robotics and Automation (ICRA), pp. 6240–6246. doi:[10.1109/ICRA46639.2022.9812082](https://doi.org/10.1109/ICRA46639.2022.9812082).
- [13] Iovino, M., Scukins, E., Styrud, J., Ögren, P., Smith, C., 2022. A survey of behavior trees in robotics and ai. *Robotics and Autonomous Systems* 154, 104096. URL: <https://www.sciencedirect.com/science/article/pii/S0921889022000513>, doi:<https://doi.org/10.1016/j.robot.2022.104096>.
- [14] Jain, A., Kemp, C.C., 2008. Behaviors for robust door opening and doorway traversal with a force-sensing mobile manipulator, in: RSS Manipulation Workshop: Intelligence in Human Environments, Georgia Institute of Technology, Zurich. URL: <http://hdl.handle.net/1853/37399>.
- [15] Jain, A., Kemp, C.C., 2010. Pulling open doors and drawers: Coordinating an omni-directional base and a compliant arm with equilibrium point control, in: 2010 IEEE International Conference on Robotics and Automation, pp. 1807–1814. doi:[10.1109/ROBOT.2010.5509445](https://doi.org/10.1109/ROBOT.2010.5509445).
- [16] Jang, K., Kim, S., Park, J., 2023. Motion planning of mobile manipulator for navigation including door traversal. *IEEE Robotics and Automation Letters* 8, 4147–4154. doi:[10.1109/LRA.2023.3279612](https://doi.org/10.1109/LRA.2023.3279612).
- [17] Johnson, M., Bradshaw, J.M., Feltovich, P.J., Jonker, C.M., van Riemsdijk, M.B., Sierhuis, M., 2014. Coactive design: Designing support for interdependence in joint activity. *J. Hum.-Robot Interact.* 3, 43–69.
- [18] Johnson, M., Shrewsbury, B., Bertrand, S., Calvert, D., Wu, T., Duran, D., Stephen, D., Mertins, N., Carff, J., Rifenburgh, W., Smith, J., Schmidt-Wetekam, C., Faconti, D., Gruber-Tilton, A., Eyslette, N., Meier, T., Kalkov, I., Craig, T., Payton, N., McCrary, S., Wiedebach, G., Layton, B., Neuhaus, P., Pratt, J., 2017. Team ihmc's lessons learned from the darpa robotics challenge: Finding data in the rubble. *Journal of Field Robotics* 34, 241–261.
- [19] Johnson, M., Vignati, M., Duran, D., 2020. Understanding Human-Machine Teaming through Interdependence Analysis. CRC Press. p. Chapter 9.
- [20] Jorgensen, S.J., Lanigan, M.W., Bertrand, S.S., Watson, A., Altemus, J.S., Askew, R.S., Bridgwater, L., Domingue, B., Kendrick, C., Lee, J., Paterson, M., Sanchez, J., Beeson, P., Gee, S., Hart, S., Quispe, A.H., Griffin, R., Lee, I., McCrary, S., Sentis, L., Pratt, J., Mehling, J.S., 2019. Deploying the nasa valkyrie humanoid for ied response: An initial approach and evaluation summary, in: 2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids), pp. 1–8. doi:[10.1109/Humanoids43949.2019.9034993](https://doi.org/10.1109/Humanoids43949.2019.9034993).
- [21] Kang, G., Seong, H., Lee, D., Shim, D.H., 2024. A versatile door opening system with mobile manipulator through adaptive position-force control and reinforcement learning. *Robotics and Autonomous Systems* 180, 104760. URL: <https://www.sciencedirect.com/science/article/pii/S0921889024001441>, doi:<https://doi.org/10.1016/j.robot.2024.104760>.
- [22] Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A.C., Lo, W.Y., Dollár, P., Girshick, R., 2023. Segment anything. arXiv:2304.02643.
- [23] Koolen, T., Bertrand, S., Thomas, G., de Boer, T., Wu, T., Smith, J., Englsberger, J., Pratt, J., 2016. Design of a momentum-based control framework and application to the humanoid robot atlas. *International Journal of Humanoid Robotics* 13, 1650007.
- [24] Marion, P., Fallon, M., Deits, R., Valenzuela, A., Pérez D'Arpino, C., Izatt, G., Manuelli, L., Antone, M., Dai, H., Koolen, T., Carter, J., Kuindersma, S., Tedrake, R., 2017. Director: A user interface designed for robot operation with shared autonomy. *Journal of Field Robotics* 34, 262–280.
- [25] Mishra, B., Calvert, D., Bertrand, S., McCrary, S., Griffin, R., Sevil, H.E., 2021. Gpu-accelerated rapid planar region extraction for dynamic behaviors on legged robots, in: 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 8493–8499.
- [26] Mishra, B., Calvert, D., Bertrand, S., Pratt, J., Sevil, H.E., Griffin, R., 2024. Efficient terrain map using planar regions for footprint planning on humanoid robots, in: 2024 IEEE International Conference on Robotics and Automation (ICRA), pp. 8044–8050. doi:[10.1109/ICRA51747.2024.10610879](https://doi.org/10.1109/ICRA51747.2024.10610879).
- [27] Moog Inc., 2023. Moog Microhydraulics Integrated Smart Actuator Datasheet. Technical Report. Moog Inc. URL: <https://www.moog.com/content/dam/moog/literature/ICD/Moog-Microhydraulics-Integrated-Smart-Actuator-Datasheet-en.pdf>. accessed: 2024-05-13.
- [28] Pettinger, A., Alambeigi, F., Pryor, M., 2022. A versatile affordance modeling framework using screw primitives to increase autonomy during manipulation contact tasks. *IEEE Robotics and Automation Letters* 7, 7224–7231. doi:[10.1109/LRA.2022.3181732](https://doi.org/10.1109/LRA.2022.3181732).
- [29] Pettinger, A., Elliott, C., Fan, P., Pryor, M., 2020. Reducing the teleoperator's cognitive burden for complex contact tasks using affordance primitives, in: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 11513–11518. doi:[10.1109/IROS45743.2020.9341576](https://doi.org/10.1109/IROS45743.2020.9341576).
- [30] Polycam, 2023. Polycam: 3d scanning made simple. <https://polycam.com/>. Accessed: 2024-05-31.
- [31] Pratt, J., Carff, J., Drakunov, S., Goswami, A., 2006. Capture point: A step toward humanoid push recovery, in: 2006 6th IEEE-RAS International Conference on Humanoid Robots, pp. 200–207. doi:[10.1109/ICHR.2006.321385](https://doi.org/10.1109/ICHR.2006.321385).
- [32] Pratt, J.E., Bertrand, S., Koolen, T., 2019. Stepping for Balance Maintenance Including Push-Recovery. Springer Netherlands, Dordrecht. pp. 1419–1466. URL: https://doi.org/10.1007/978-94-007-6046-2_41, doi:[10.1007/978-94-007-6046-2_41](https://doi.org/10.1007/978-94-007-6046-2_41).
- [33] Roboflow, 2023. Yolov8 dataset. <https://universe.roboflow.com/test-bwgub/yolov8-il19q>. URL: <https://universe.roboflow.com/test-bwgub/yolov8-il19q>. visited on 2024-04-27.
- [34] SAKE Robotics, 2024. Ezgripper™ robotic grippers. <https://sakerobotics.com/>.
- [35] StereoLabs, 2024. Zed 2 - ai stereo camera | stereolabs. URL: <https://www.stereolabs.com/products/zed-2>. accessed: 2024-04-26.
- [36] Xiong, H., Mendonca, R., Shaw, K., Pathak, D., 2024. Adaptive mobile manipulation for articulated objects in the open world. arXiv:2401.14403.
- [37] Zhang, M., Ma, Y., Miki, T., Hutter, M., 2024. Learning to open and traverse doors with a legged manipulator. URL: <https://arxiv.org/abs/2409.04882>, arXiv:2409.04882.

Fast Humanoid Robot Door Traversals



Duncan Calvert received a B.S. in Computer Science at the University of West Florida (UWF) in 2014. He has worked as a Research Associate in the Robotics Lab at IHMC since 2014. He served as the operator's copilot in the DARPA Robotics Challenge Finals in 2015, where IHMC took 2nd place. He is now a PhD student in the Department of Intelligent Systems and Robotics (ISR) at UWF. His research interests include legged locomotion, manipulation, behaviors, and teleoperation for humanoid robots.



Luigi Penco is a research scientist at IHMC and focuses on developing autonomous and semi-autonomous behaviors for humanoid robots, as well as user interfaces for their control. He earned a bachelor's degree in Electronics Engineering from Roma Tre University in 2015 and a master's in Artificial Intelligence and Robotics from La Sapienza University of Rome in 2018. In 2022 he received a PhD in robotics from Université de Lorraine, while conducting his doctoral studies at Inria Nancy Grand-Est. His research interests include teleoperation systems, impact-aware control strategies and behavior generation, coordination and adaptation methods.



Dexton Anderson is a Software and DevOps Engineer working with the humanoid robotics team at IHMC. He is from Pensacola and received his BSc. in Computer Science from UWF. He contributes to various IHMC vendored software libraries used by the humanoid robotics platforms IHMC has access to and maintains continuous integration systems used by the robot lab. Dex has previously worked for The National Flight Academy (NAS Pensacola) developing flight simulations and add-ons for Prepar3D flight simulator. He also has created and contributes to many open-source software projects in his free time.



Tomasz Bialek received his B.S. in Computer Science at the University of West Florida in 2024. He has been an intern at IHMC since 2023. He has worked on humanoid robot hand low level control and user interfaces, GPU acceleration of sensor data acquisition, and heuristic object detection algorithms that combine the results of multiple state of the art approaches. He enjoys photography and working on perception algorithms.

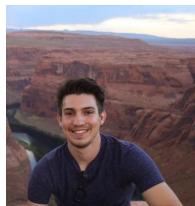


Arghya Chatterjee received a B.S. in Mechanical Engineering (Robotics Major) at the Bangladesh University of Engineering and Technology (BUET) in 2019. He has worked as a software engineer in Lotus Robotics where he developed autonomous UGVs for industrial security inspection and food delivery since 2020. He was a perception team member of Team COSTAR from NASA JPL during DARPA SubT Challenge where he worked on degeneracy-aware SLAM, Object Detection and Localization techniques for Single and Multi-robot autonomous systems in 2020-2021. He is now a PhD student in the Department of Intelligent Systems and Robotics (ISR) at UWF.

His research interests include object detection, pose estimation, object surface reconstruction and autonomous manipulation for humanoid robots.



Bhavyansh Mishra received a B.S. in Computer Engineering at the University of West Florida (UWF) in 2019. During his undergraduate degree, he also published research on DevOps, distributed software architectures, and secure software development practices. He has also received numerous awards in computer vision, cybersecurity, and autonomous drone racing competitions. He also defended his PhD dissertation on perceptive bipedal locomotion in the Department of Intelligent Systems and Robotics (ISR) in 2024. His research focused on perception and planning algorithms that enabled bipedal humanoid robots to traverse challenging terrain in real-world applications.



Geoffrey Clark is a Research scientist at IHMC and focuses on machine learning and artificial intelligence research for humanoid robots. In 2023 he received a PhD in Electrical Engineering from Arizona State University, while pursuing research in the Human Robot Interaction Lab. During his doctoral research, Dr. Clark created generative probabilistic models of legged walking systems, including autonomous robots, prosthetics, and exoskeletons. Using these models he examined how to utilize the probabilistic framework to optimize human and robot interactions for beneficial outcomes, such as reducing joint loads of amputees to reduce the risk of osteoarthritis.



Sylvain Bertrand received a PhD in Mechanical Engineering at the University of Versailles in 2013. He has worked as Research Scientist at the Robotics Lab at IHMC since 2013. He was the control lead during the DARPA Robotics Challenge from 2013 to 2015, where IHMC took 2nd place in the finals. He has since contributed in several domains such as: whole-body controls, perception, manipulation, and teleoperation for humanoid robots.



Robert Griffin is a Research Scientist with IHMC, where he leads the Robotics team. His research focuses on improving mobility and autonomy for legged robotics and powered exoskeletons. He is interested in system level approaches for improving the mobility and capability of these robotic platforms, including platform design, motion design and control, autonomy and manipulation, and perception. His work involves leading a number of projects, including developing the next generation humanoid robot, Nadia, which has a high power-to-weight ratio and large range of motion and will enable exploration and interaction with complex, dynamic, and unknown environments such as ships and urban settings. Robert received his PhD from Virginia Tech in 2017.