2020 IEEE International Conference on Robotics and Automation (ICRA)
31 May - 31 August, 2020. Paris, France

Finding Locomanipulation Plans Quickly in the Locomotion Constrained Manifold

Steven Jens Jorgensen[1,2], Mihir Vedantam[3], Ryan Gupta[3], Henry Cappel[3], and Luis Sentis[3]

***Abstract*— We present a method that finds locomanipulation plans that perform simultaneous locomotion and manipulation of objects for a desired end-effector trajectory. Key to our approach is to consider an injective locomotion constraint manifold that defines the locomotion scheme of the robot and then using this constraint manifold to search for admissible manipulation trajectories. The problem is formulated as a weighted-A* graph search whose planner output is a sequence of contact transitions and a path progression trajectory to construct the whole-body kinodynamic locomanipulation plan. We also provide a method for computing, visualizing, and learning the locomanipulability region, which is used to efficiently evaluate the edge transition feasibility during the graph search. Numerical simulations are performed with the NASA Valkyrie robot platform that utilizes a dynamic locomotion approach, called the divergent-component-of-motion (DCM), on two example locomanipulation scenarios.**

I. INTRODUCTION

To exploit the full capabilities of humanoid robots in human-centered environments, it is critical that the robots are able to efficiently interact with objects designed for human use. However, much of the success with locomanipulation of objects has been seen with wheeled-based mobile-manipulators [1], [2], [3], [4]. For instance, [4] shows robust manipulation of kinematically constrained objects such as doors and cabinets. The success of wheeled-bases is unsurprising as the manifold for locomotion and manipulation is continuous which simplifies the search for feasible plans. However, robots with limbs rely on contact transitions to perform locomotion. As breaking and making contacts are discrete decisions that introduce discontinuity and can even be combinatorial when finding an appropriate contact mode schedule [5], it is non-trivial to identify a sequence of dynamically feasible contact transitions during manipulation.

One way to address the discontinuity issue with coupled locomotion and manipulation of limbed robots is to treat the floating degrees of freedom (DoF) of the robot to be controllable, for instance by constraining it to $SE(2)$, then solving the locomanipulation problem as one would with a wheeled-base robot and finding a satisfying quasi-static sequence of footsteps [6]. A more recent approach treats the end-to-end locomanipulation problem as rearrangement planning, however it also only outputs quasi-static solutions [7]. The difficulty of handling contact transitions while performing manipulation is the reason that whole-body manipulation of objects by limbed robots are often performed by maintaining the same stance configuration throughout the entire manipulation trajectory. For example, in [8], bi-manual manipulation of a humanoid robot is performed with the same stance configuration. In [9], locomotion, locomanipulation, and manipulation zones are constructed to approach the object in the manipulation zone. Here, the manipulation task is also performed with a fixed stance. Furthermore, all the previously mentioned approaches are only able to output quasi-static solutions.

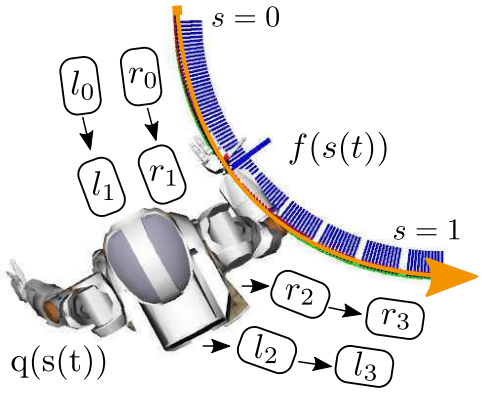

Fig. 1. A top-view visualization of the considered locomanipulation problem definition. Given a manipulation constraint end-effector path/s described by $f(s)$, the goal is to find a progression trajectory $s(t) \in [0, 1]$, with t being time, and a sequence of contact transitions (l_i, r_i) such that the resulting whole-body trajectory $q(s(t))$ also satisfies the prescribed locomotion manifold. A solution is a feasible locomanipulation plan.

In contrast, we present an approach that is able to find dynamic locomanipulation plans with kinodynamic whole-body solutions. This is done by first defining the locomotion constraint manifold and then finding manipulation plans that satisfy the original locomotion constraint. This is equivalent to finding manipulation trajectories in the *nullspace* of the locomotion. As a motivating example, we consider the locomotion constraint manifold to be the task-space trajectories generated by the dynamic locomotion approach called the divergent-component-of motion (DCM) [10] that is used on the NASA Valkyrie robot [11] with a momentum-based whole-body controller [12]. Note that our approach is invariant to an injective locomotion scheme (see Sec. II-B) which results to producing kinodynamic trajectories automatically if the locomotion scheme is dynamic. Additionally, if the locomotion approach has stability properties, the resulting whole-body trajectories will also have these properties.

Next, we formulate locomanipulation as the following problem. Given an $SE(3)$ end-effector trajectories for the

[1]The author is supported by a NASA Space Technology Research Fellowship (NSTRF). The authors are also with the [2]Department of Mechanical Engineering and [3]Department of Aerospace Engineering and Engineering Mechanics in the University of Texas at Austin

hands, the goal is to find a progression trajectory for the hands with a satisfying sequence of footsteps such that the resulting whole-body trajectory also satisfies the locomotion constraint manifold (See Fig. 1). We solve this as a low-dimensional graph search problem with a weighted A* as the planner. To efficiently compute feasible edge transitions that can be manipulation, locomotion, or locomanipulation trajectories, we introduce a method for learning the locomanipulability regions of the robot with the prescribed locomotion constraint manifold with a neural-network based classifier. The solution of the planner is a kinematically feasible trajectory that respects joint limits. Kinodynamic satisfiability is also achieved if the external disturbance of the manipulation task can be sufficiently rejected or compensated by the low-level whole-body controller. Finally, we show that we are able to generate fast locomanipulation plans on two example problems.

Our paper has two key contributions. First, we introduce a novel method to compute, visualize, and learn the locomanipulability regions, defined as the region in which both manipulation and locomotion are possible. Second, we introduce a fast weighted A* planner formulation which uses the learned locomanipulation regions to find satisfying locomanipulation plans.

A. Related Works on Locomanipulation

While the problem of finding locomanipulation plans is discussed here, there are other recent works on locomanipulation-related problems such as [13], [14], [15], [16]. In [13], a taxonomy of locomanipulation poses is presented as well as an example analysis of required pose transitions to climb stairs. [15] provides a ground work for understanding environment affordances for locomanipulation. [16] extends [13] and [15] by using data to auto-generate a pose transition graph and testing their affordance classifications on a mobile manipulator with a wheeled base.

We previously described existing quasi-static approaches that used search based algorithms to solve locomanipulation problems. However, our idea of dynamic locomanipulation by finding manipulation trajectories in the nullspace of locomotion has been previously pursued in [17]. In their work, primitives for both locomotion and manipulation are generated beforehand. Then, an offline RRT-based planner is used to find locomanipulation plans in the intersection of the primitives' image spaces. Our work differs from them in a few ways. First, we have a different problem and planner formulation for finding locomanipulation plans. For instance, we consider manipulating objects with predefined manipulation trajectories (e.g. as described by affordance templates (ATs) [18]). Next, because their method consists of a search over the nullspace of the prioritized motion primitive, pure locomotion or pure manipulation phases are not considered in their framework, which is not a limitation in our planner. We believe our planner formulation is faster as their search approach is performed offline for an unspecified amount of computation time. Another work, [19], uses a search based algorithm for planning contact transitions for the purposes of locomotion and manipulation for many types of robots. However, the coupled locomotion and manipulation problem are not considered. More recently, [20] presents a method for addressing the coupled locomotion and manipulation problem as we do here. However, their results are limited in two ways. First, they operate on low-dimensional (DoF) systems while we solve our locomanipulation problem with a full humanoid. Second, they do not consider joint limits, while our approach automatically handles kinematic constraints. A complete kinodynamic planner utilizing SQP methods was presented in [21], but it is prohibitively expensive and requires good initial conditions with computation times on the order of several minutes. In contrast, our problem formulation outputs candidate solutions in less than 5 seconds and complete validated solutions within one minute.

II. APPROACH OVERVIEW

To find locomanipulation plans, the key idea is to first consider that the locomotion scheme for the robot is provided ahead of time. This constrains the possible locomotion trajectories that the robot can execute. Then, locomanipulation is achieved by finding admissible manipulation trajectories that satisfy both the original locomotion constraint and the desired manipulation end-effector trajectory. We consider limbed robots of humanoid form, but the ideas presented here can also work with other multi-limbed robots.

A. Problem Definition

The locomanipulation problem is formulated as follows: given a desired end-effector path trajectory, $f(s)$, the goal is to find a manipulation progression variable trajectory for $s \in [0, 1]$ as a function of time, t, which defines $s(t)$, and a footstep sequence trajectory such that the resulting whole-body trajectory $q(s(t))$ satisfies the end-effector path trajectory $f(s(t))$ and the locomotion constraint manifold. For instance, suppose the robot's task is to open a door (See Fig. 1). The desired end-effector trajectory for the hand can be defined in terms of the trajectory of the handle as the door opens. This is similar to how ATs [18] or task space regions (TSRs) [8] would define the robot interface to the door. At any point in time, the robot may decide to pull on the door, take a footstep, or do both at the same time. For example, an action which pulls the door is a progression of the s variable from s_i to s_{i+1}. We call this an increment of the manipulation variable by some Δs.

B. Defining the Locomotion Constraint Manifold

Existing locomotion schemes in limbed robots for example are performed with quasi-static, capture-point [22], divergent component of motion (DCM) [10], time-velocity-reversal (TVR) [23], or centroidal-momentum based planners [24]. These high-level planners output center-of-mass (CoM) trajectories (and sometimes momentum trajectories) for a given sequence of contact modes. Consequently, to satisfy these centroidal trajectories with contact constraints, task space trajectories for the end-effectors such as the feet, palm, pelvis, etc, also have to be constructed by an accompanying planner.

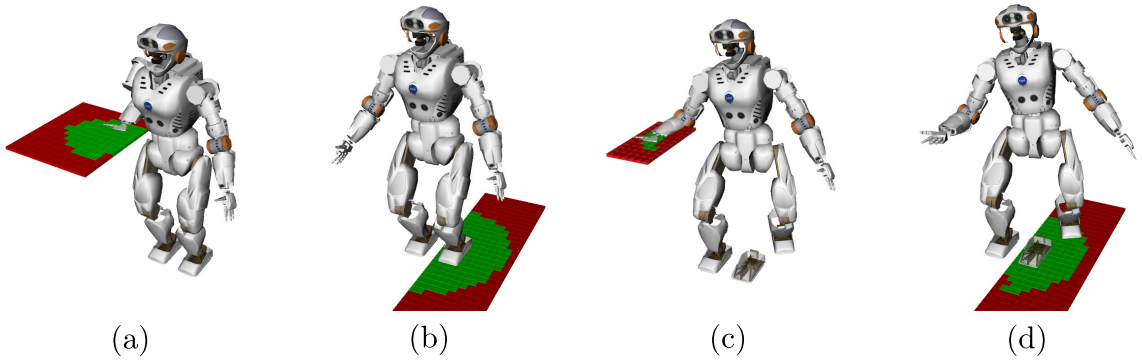

Fig. 2. A visualization of (a) the manipulation reachability region of the right hand, (b) the locomotion contact transition reachability region, (c) the locomanipulation region in the hand end-effector space, and (d) the locomanipulation region in the contact transition space. Green and red regions indicate valid and invalid regions respectively. In (a) and (b), valid regions are reachable end-effector poses and contact transitions from the initial configuration respectively. For (c) and (d), valid regions indicate the locomanipulability region from the initial configuration. The transparent left foot in (c) and (d) indicates the starting stance. In (c), the locomanipulability region is the region in which different hand poses can be kinematically maintained while the same contact transition is performed. While in (d), the locomanipulablity region is the region in which different foot contact transitions can be performed while maintaining a fixed hand pose. Reachability and locomanipulability regions are generally in 3D, but the above visualizations are only performed on a 2D slice. Notice that the locomanipulability regions are always a subset of the corresponding reachability regions, i.e. (c) $\subseteq$ (a) and (d) $\subseteq$ (b).

Additionally, these high-level planners which constitute the locomotion scheme are typically injective. That is, for a given sequence of contact modes and an initial condition of the robot configuration q, $\dot{q}$, it will always output the same task space trajectories, $x(t)$ for the CoM and end-effectors. For humanoid walking these task space trajectories could be

$$x_{\mathrm{L}}(t) = [x_{\mathrm{COM}}(t), x_{\mathrm{foot}}^{\mathrm{left}}(t), x_{\mathrm{foot}}^{\mathrm{right}}(t), x_{\mathrm{pelvis}}(t)]^T, \quad (1)$$

with the task spaces defined such that $x_{\mathrm{COM}} \in \mathbf{R}^3$, $x_{\mathrm{foot}}(t) \in SE(3)$, and $x_{\mathrm{pelvis}} \in SO(3)$. Additionally these tasks will have corresponding locomotion task Jacobian,

$$\Delta x_{\mathrm{L}}(t) = J_{\mathrm{L}}(q(t))\Delta q(t) \quad (2)$$

Thus, the locomotion scheme provides a constraint manifold, Eq. (2), that needs to be satisfied when finding admissible manipulation plans.

C. Defining the Locomanipulability Region

We define the locomanipulability region to be the area in which both locomotion and manipulation tasks are feasible. By constraining the locomotion scheme, we are able to test whether a particular manipulation trajectory (e.g. a hand end-effector trajectory) satisfies a given locomotion manifold. Equivalently, a manipulation constraint can be initially set and used to check whether the original locomotion plan is still valid. When both manipulation and locomotion trajectories are feasible, locomanipulation becomes possible. From the problem definition, the manipulation constraint can be described in terms of s, namely:

$$\Delta x_{\mathrm{M}}(s) = J_{\mathrm{M}}(q(s))\Delta q(s), \quad (3)$$

where the subscript M indicates manipulation tasks in SE(3) with its corresponding Jacobian.

Numerically checking whether a manipulation trajectory $x_{\mathrm{M}}(s)$ is admissible for a given locomotion manifold $x_{\mathrm{L}}(t)$ is checked with a series of inverse-kinematics (IK) that simulate the whole-body controller on the robot (See Sec. III-B).

Similar to reachability regions [25] for manipulation (Fig. 2a) and locomotion (Fig. 2b), we can define the locomanipulability region as a region in space for which both locomotion and manipulation tasks are possible. This region can be defined either in the end-effector space (Fig. 2c) or the contact transition space (Fig. 2d). For the former, if the contact transition is fixed (ie: the robot is set to take a left footstep), there will only be a small region in the end-effector space for which manipulation trajectories are possible. For the latter, suppose the robot's right hand is to be constrained in a particular pose in $SE(3)$, then the region on the floor for which footstep transitions are possible will be the locomanipulation region defined in the contact transition space. (Fig. 2d).

III. IMPLEMENTATION DETAILS[1]

A. Locomotion Manifold Parameters

The following task space trajectories for CoM, feet and pelvis are based on a simplified behavior of The Institute for Human Machine & Cognition's (IHMC) walking controller on NASA's Valkyrie robot. For a given foot contact sequence and initial condition of the CoM state, the DCM generates a CoM trajectory based on a specified swing foot time, double support transfer time, and final settling time. At the beginning and end of the DCM trajectory, the desired virtual repellant point (VRP) is set at the support polygon center, so that the beginning and ending of each walking trajectory will have the CoM at the support polygon center. In addition to the CoM trajectory, satisfying task space trajectories for the feet and pelvis still need to be set. Throughout the walking trajectory the pelvis orientation is always the average of the orientation of the feet using spherical linear interpolation (SLERP)[26].

[1]https://github.com/stevenjj/icra2020locomanipulation

The average of the feet orientation and position is referred to as the midfeet frame.

$$x_{\text{pelvis}}(t) = \text{SLERP}(0.5, x_{\text{foot}}^{\text{left}}(t), x_{\text{foot}}^{\text{right}}(t)). \tag{4}$$

If at the start of the DCM trajectory the pelvis orientation is not equal to midfeet frame orientation due to manipulation tasks, a hermite quaternion curve [27] is used to interpolate the pelvis orientation before the robot begins to walk.

For the swing foot position, We use two hermite curves with boundary conditions at the apex of the foot swing. At the apex of the swing, the velocity of the foot is set to be the average velocity of the swing foot defined as

$$\dot{x}_{\text{foot}}(\frac{t_{\text{swing}}}{2}) = \frac{\Delta x_{\text{foot}}}{t_{\text{swing}}}, \tag{5}$$

where Δx_{foot} is the total distance traveled by the swing foot and t_{swing} is the swing time. The swing foot orientation is constructed with a single hermite quaternion curve with zero angular velocity boundary conditions. Finally, if the foot is in stance or in double support, its position and orientation are held constant.

B. IK Configuration Trajectory

For a given desired locomotion and manipulation task space trajectories, a feasible IK trajectory with these two tasks simultaneously implies that the desired locomanipulation trajectories are feasible. For a given footstep contact sequence, we obtain a locomotion task space trajectory $x_{\text{L}}(t)$ with duration ΔT. Similarly, for a given increment of the manipulation variable, Δs, we obtain a manipulation task trajectory $x_{\text{M}}(s)$. The locomotion and manipulation trajectories can be parameterized by an indexing variable $i \in \{0, 1, ..., N\}$, a discretization factor N, and making the following substitutions

$$t(i) = t_o + \frac{i\Delta T}{N}, \tag{6}$$

$$s(i) = s_o + \frac{i\Delta s}{N}, \tag{7}$$

with t_o and s_o the initial values of t and s at $i = 0$. We can then create the locomanipulation task by stacking the tasks and their Jacobians with $x_{\text{LM}}(i) = [x_{\text{L}}^T(i), x_{\text{M}}^T(i)]^T$ and $J_{\text{LM}}(i) = [J_L^T(i), J_M^T(i)]^T$. We also add a posture joint position task J_P with task errors Δx_P in the the torso which helps condition the trajectories to be near a desired nominal pose. Then, the IK configuration trajectory, which mirrors the controller behavior of the robot, is performed using the following equations.

$$\Delta x_{\text{LM}}(i) = x_{\text{LM}}(i) - x_{\text{LM}}(q(i)), \tag{8}$$

$$\Delta q(i) = k_p \cdot \overline{J}_{\text{LM}}(i)\Delta x_{\text{LM}}(i) + \overline{(J_P N_{\text{LM}})}(\Delta x_P), \tag{9}$$

$$q(i+1) = c(q(i) + \Delta q(i)), \tag{10}$$

where $\overline{X} = (A^{-1}X^T)(XA^{-1}X^T)^\dagger$ is the dynamically consistent pseudoinverse [28] with A being the inertia matrix for a robot configuration $q(i)$ and † indicates the pseudoinverse. $N_{\text{LM}} = (I - \overline{J_{\text{LM}}}J_{\text{LM}})$ is the nullspace of the locomanipulation task, with I the identity matrix. The task error at the i-th index is defined by Eq. (8) in which $x_{\text{LM}}(q(i))$ is the current task space poses given the robot configuration. The configuration change is obtained using Eq. (9) with k_p a scalar gain, and a configuration update is performed with Eq. (10) with $c(\cdot)$ being a clamping function that ensures joint limits are not exceeded. Finally, Eqs. (8)-(10) are iteratively repeated. If an iteration causes the task error to increase, backtracking on k_p is performed by updating it with $k_p^* = \beta k_p$ with $\beta = 0.8$. The trajectory converges when all the $\Delta x_{LM}(i)$ are driven to 0. The trajectory fails to converge when the norm of $\Delta q(i)$ goes below 1e-12.

TABLE I

CLASSIFIER FEATURE VECTOR

Type	Feature Name	Dim
$\mathbf{R}^1$	Stance Leg	1
$\mathbf{R}^1$	Manipulation Type	1
$SE(3)$	Pelvis Starting Pose	6
$SE(3)$	Swing Start and Land Foot Pose	12
$SE(3)$	Left and Right Hand Poses	12

TABLE II

EDGE FEASIBILITY CHECK WITH AN INTEL I5-9600K CPU

Transition feasibility check type	Time per edge (seconds)
IK Trajectory	(2.11 $\pm$ 0.13)
Neural Network Classifier	(1.44 $\pm$ 0.18) e $-$ 3

C. Learning the Locomanipulability Region

When deciding whether or not a contact transition and a progression variable Δs change is possible, instead of running the full IK trajectory to check for convergence, we instead learn a classifier that learns the result of the IK trajectory for the given task space inputs. Similar to the approach presented in [29] that used a neural network for classifying contact transition feasibility, the classifier used here will learn the trajectory feasibility but instead with a manipulation constraint. The classifier is a 3-layer fully connected network with 100 ReLu units per layer [30] and a sigmoid activation function for binary classification. The network is trained with the keras framework [31].

The input vector, $p(v_1, v_2; s)$, used for the neural network classifier can be seen in Table I. The input vector is a function of the two graph vertices (v_1, v_2) as described in Sec. III-D, but it is parameterized by the location of the end-effector along the manipulation trajectory, $f(s)$. The stance leg is a binary variable that indicates which leg is the stance leg (left, right). Similarly, the manipulation type indicates the manipulator end-effectors (left, right, or both hands). The remainder of the features are the 6D poses of the specified robot body part with respect to the stance foot. As there are two choices for the swing leg and three choices for manipulation type, there are six possible contact transitions to consider. For each contact transition type, the training data is generated by randomly generating the upper body joint configurations, and randomly selecting a foot landing location w.r.t the stance foot as the origin. The pelvis pose is also randomly generated in the convex hull

of the feet. For a particular manipulation type, we fix the manipulator pose and solve a series of IKs (Sec. III-B) that simulate the robot's whole body controller to check if the locomanipulation trajectory is feasible.

The output of the classifier is a prediction score, $y(\cdot) \in [0,\ 1]$, that indicates the feasibility of the queried transition. Since the classifier is only trained on data that represents locomanipulation with a fix manipulator pose ($\Delta s = 0$), additional steps are taken to use the classifier for manipulation-only decisions and locomanipulation decisions with a moving manipulator pose ($\Delta s \neq 0$). When considering the manipulation only case, the manipulation trajectory is discretized into N_m equidistant points and a step in place trajectory is queried from the neural network for each point. This method assumes that if the discretized points are in the locomanipulation region then the entire trajectory must be as well. The lowest score is then taken as the feasibility score. For the locomanipulation with a moving manipulator pose case ($\Delta s \neq 0$), a similar discretization is used but instead of testing a step in place, the specified footstep from v_1 to v_2 is tested at each of the points. Once again, the lowest score is taken as the feasibility score. A succinct description for the feasibility score is written in Eq. (11) below. Table II shows that the classifier evaluates edge transitions more than three orders of magnitude faster than an IK approach on a CPU-based implementation only.

$$n(v_1, v_2) = \begin{cases} y(p(v_1, v_2; s)), & \Delta s = 0 \\ \min\limits_{i=1,\ldots,N_m} y(p(v_1, v_2; s_i)), & \Delta s \neq 0. \end{cases} \tag{11}$$

D. Weighted A* Formulation

Finding locomanipulation plans is formulated as a low-dimensional graph search problem, $G = (V, E)$. Each vertex $v \in V$ is a locomanipulation state $v = (s, x_{\text{feet}}, y_{\text{feet}}, \theta_{\text{feet}}) \in \mathbf{R}^7$, where s is the manipulation variable state, and $(\cdot)_{\text{feet}}$ are the states of the left and right feet. The states are discretized from the starting position of the robot. We assume that the starting position of the robot with $f(s = 0)$ is such that the configuration is in the locomanipulation region. Only a finitely sized lattice is considered by defining a kinematic reachability limit from a certain radius (e.g. 1.5m) from $f(s)$. An edge $e \in E$ in the graph is a transition between two vertices v_1 and v_2 which can have a Δs change, which progresses the manipulation variable, and/or a foot contact transition. This enables the planner to make a decision between performing manipulation, locomotion, or locomanipulation trajectories.

1) Edge Cost: A contact transition between two vertices has the following edge transition cost.

$$\begin{aligned} \Delta g(v_1, v_2) =& w_s \cdot (1 - s) + w_{\text{step}} + \\ & w_{\text{L}} \cdot r(v_2) + w_d \cdot (1 - n(v_1, v_2)), \end{aligned} \tag{12}$$

where w_s encourages the progression of the manipulation trajectory, w_{step} is a scalar cost of taking a footstep, w_L penalizes states that deviate from a suggested body path $r(v_2)$, and w_d penalizes edge transitions that have low feasibility computed with $n(v_1, v_2)$. If the classifier is not used, w_d is zero. The suggested body path can be an output from the same high-level planner that produced the end-effector trajectory for $f(s)$. Here we first compute $T_{\text{foot}}^{s_0}$, which is the fixed transform between the initial end-effector pose, $f(s = 0)$, and the starting stance foot pose. For a given s, we then transform the initial stance to the corresponding pose of $f(s)$ using $T_{\text{foot}}^{s_0}$. Then $r(v_2)$ is computed as the norm of the difference between a foot landing location in v_2 and the aforementioned transformation.

Since the planner is successful when it finds a feasible path to a state such that $s = 1$, notice that maximizing for feasibility is not necessarily the best course of action as the planner can mindlessly perform contact transitions that are feasible. Therefore, a trade-off has to be performed between progressing the manipulation variable, s, attempting a transition using the suggested body path $r(v_2)$, deciding whether or not to make a footstep transition at all, or choosing a vertex that maximizes for feasibility.

2) Edge Transition Feasibility: To increase efficiency when not using the classifier, all neighbors are assumed to be feasible until the vertex is extracted from the priority queue. When the assumed feasible vertex is extracted from the queue, edge validity is performed by testing if a feasible transition exists from v_1 to v_2 by solving the IK configuration trajectory between two vertices. If the edge between v_1 and v_2 is not feasible, the next vertex in the priority queue is processed. On the other hand, if the classifier is used, neighbors undergo an edge feasibility check. A feasibility score of $n(v_1, v_2) > 0.5$ is labeled as a valid edge, while edges with lower scores are pruned from the graph.

3) Graph Search: The weighted A* is used as the planner [32] for the graph search problem to produce sub-optimal but faster plans than A*. The following heuristic $h(v)$ with scalar weight w_h brings s to 1 with

$$h(v) = w_h \cdot (w_s(1 - s)). \tag{13}$$

When $w_h = 1$, the solution of the planner is the optimal result produced by the A* as the heuristic is admissible [33] since Eq. (13) will be equal to the first term of Eq 12. Similar to [29], we use an ϵ-greedy strategy [34] to aid escaping cul-de-sac scenarios by randomly evaluating a vertex in the priority queue with probability ϵ $(0 < \epsilon < 1)$.

Note that the classifier-based planner can make mistakes on feasible and unfeasible transitions. Thus, when it outputs a candidate solution consisting of a sequence of edge transitions, a reconstruction step is performed in which the full IK trajectory is computed to validate the candidate plan. A complete plan is returned only if the candidate plan converged. If it did not converge, the A* planner is re-run again with an updated table that marks the invalid edge to be infeasible and the validated edges to be feasible. The remaining edges use the transition feasibility computed by the classifier from the previous run. This table ensures that the classifier-based planner saves computation time and does not repeat the mistake. To be clear, this reconstruction step is not needed if the classifier is not used or if the classifier is trusted to capture the locomanipulation region well.

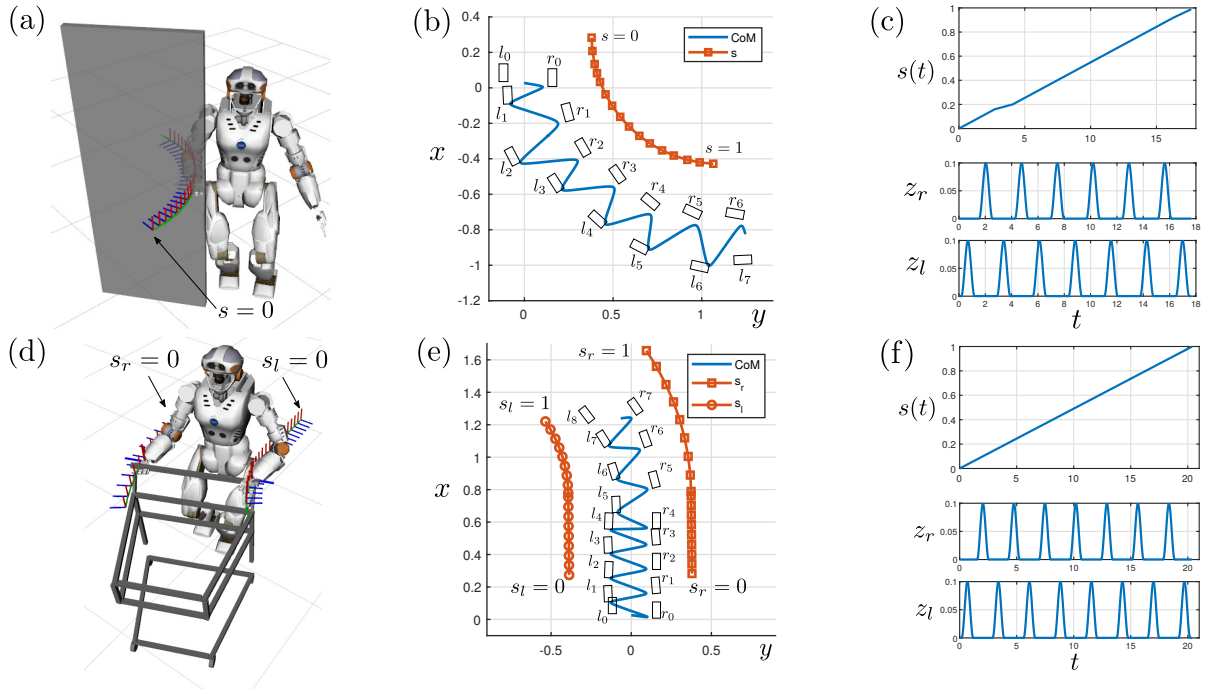

Fig. 3. A 3D view of Valkyrie opening the door (a) and pushing a cart (d). (b) and (e) show a top-view of the center-of-mass trajectory(CoM), the manipulation end-effector trajectory $f(s)$ with $s = 0$ and $s = 1$ indicating the start and ending end-effector poses respectively, and l_j and r_j indicate the j-th left and right footsteps respectively. In (d) note that $s = s_l = s_r$. (c) and (f) show the manipulation progression variable trajectory $s(t)$ as a function of time, and a visualization of the footstep contact transitions using the z-height of the left and right footsteps (z_l, z_r respectively).

TABLE III

PLANNER PERFORMANCE WITH AND WITHOUT THE CLASSIFIER

Planner Type	**Time to Goal Vertex (secs)**		**Reconstruction Time (secs)**		**Total Planning Time (secs)**	
	Door Opening	Cart Pushing	Door Opening	Cart Pushing	Door Opening	Cart Pushing
With the Classifier	**4.78s**	**3.32s**	28.31s	25.41s	**33.10s**	**28.73s**
Without the Classifier	38.29s	32.24s	0.0	0.0	38.29s	32.24s

IV. RESULTS AND DISCUSSIONS

We provide two examples in which locomanipulation is achieved for a given end-effector task space trajectory. Fig. 3 shows a figure of Valkyrie opening a door and performing a bimanual push of a cart. Trajectories are only visualized with RViz [35]. Table III shows that utilizing the locomanipulability classifier can find goal vertices or candidate plans in less than 5 seconds. While utilizing a GPU implementation can further reduce the time to generate candidate plans, the reconstruction step for confirming the full trajectory feasibility is the bottleneck as it adds 25-30s to validate the plan. Still, our approach is faster than alternative humanoid locomanipulation approaches ([17], [21]) as discussed in Sec.I-A. Additionally, a better IK implementation should decrease the overall planning time for both planner types. A criticism is that the advantage of the classifier-based planner is largely lost due to the reconstruction or validation step needed. Other works that employ a similar classifier-based approach to planning ([29], [36]) assume that the candidate plan is the final plan. Therefore, this step is optional for a well-trained or highly trusted classifier. Furthermore, the classifier's performance largely depends on how well the classifier learned the locomanipulability regions. As the classifier is used to reduce the total IK computations, the reconstruction step guarantees kinodynamic feasibility for the returned plan. Overall, we have demonstrated a fast approach for finding locomanipulation plans by finding admissible manipulation trajectories in the locomotion constraint manifold. Our approach is also invariant to an injective locomotion scheme. While our approach produces kinodynamic plans, our method relies on the user or another high-level planner to provide end-effector plans with an optional suggested body path. The full-body plans can be immediately used on the robot by using its existing API as done previously in [37], but a robust implementation would require online replanning of hand trajectories (e.g [4]) and recovery procedures if deviations in forces or kinematic trajectories have been detected due to tracking and/or modeling errors.

ACKNOWLEDGMENTS

We are grateful to Junhyeok Ahn of the Human Centered Robotics Lab for his C++ neural network code. This work is supported by a NASA Space Technology Research Fellowship (NSTRF) grant #NNX15AQ42H.

REFERENCES

[1] W. Meeussen, M. Wise, S. Glaser, S. Chitta, C. McGann, P. Mihelich, E. Marder-Eppstein, M. Muja, V. Eruhimov, T. Foote, *et al.*, "Autonomous door opening and plugging in with a personal robot," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 729–736.

[2] T. Rühr, J. Sturm, D. Pangercic, M. Beetz, and D. Cremers, "A generalized framework for opening doors and drawers in kitchen environments," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, 2012, pp. 3852–3858.

[3] T. Welschehold, C. Dornhege, and W. Burgard, "Learning mobile manipulation actions from human demonstrations," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 3196–3201.

[4] M. Arduengo, C. Torras, and L. Sentis, "A versatile framework for robust and adaptive door operation with a mobile manipulator robot," *arXiv preprint arXiv:1902.09051*, 2019.

[5] M. Posa and R. Tedrake, "Direct trajectory optimization of rigid body dynamical systems through contact," in *Algorithmic foundations of robotics X*. Springer, 2013, pp. 527–542.

[6] S. Dalibard, A. Nakhaei, F. Lamiraux, and J.-P. Laumond, "Manipulation of documented objects by a walking humanoid robot," in *10th IEEE-RAS International Conference on Humanoid Robots*, 2010, pp. pp–518.

[7] J. Mirabel and F. Lamiraux, "Constraint graphs: Unifying task and motion planning for navigation and manipulation among movable obstacles," *hal-01281348*, 2016.

[8] D. Berenson, S. Srinivasa, and J. Kuffner, "Task space regions: A framework for pose-constrained manipulation planning," *The International Journal of Robotics Research*, vol. 30, no. 12, pp. 1435–1460, 2011.

[9] P. Ferrari, M. Cognetti, and G. Oriolo, "Humanoid whole-body planning for loco-manipulation tasks," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 4741–4746.

[10] J. Englsberger, C. Ott, and A. Albu-Schäffer, "Three-dimensional bipedal walking control based on divergent component of motion," *IEEE Transactions on Robotics*, vol. 31, no. 2, pp. 355–368, 2015.

[11] N. A. Radford, P. Strawser, K. Hambuchen, J. S. Mehling, W. K. Verdeyen, A. S. Donnan, J. Holley, J. Sanchez, V. Nguyen, L. Bridgwater, *et al.*, "Valkyrie: Nasa's first bipedal humanoid robot," *Journal of Field Robotics*, vol. 32, no. 3, pp. 397–419, 2015.

[12] T. Koolen, S. Bertrand, G. Thomas, T. De Boer, T. Wu, J. Smith, J. Englsberger, and J. Pratt, "Design of a momentum-based control framework and application to the humanoid robot atlas," *International Journal of Humanoid Robotics*, vol. 13, no. 01, p. 1650007, 2016.

[13] J. Borras and T. Asfour, "A whole-body pose taxonomy for loco-manipulation tasks," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 1578–1585.

[14] E. Farnioli, M. Gabiccini, and A. Bicchi, "Toward whole-body loco-manipulation: Experimental results on multi-contact interaction with the walk-man robot," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 1372–1379.

[15] P. Kaiser, E. E. Aksoy, M. Grotz, and T. Asfour, "Towards a hierarchy of loco-manipulation affordances," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 2839–2846.

[16] T. Asfour, J. Borràs, C. Mandery, P. Kaiser, E. E. Aksoy, and M. Grotz, "On the dualities between grasping and whole-body loco-manipulation tasks," in *Robotics Research*. Springer, 2018, pp. 305–322.

[17] A. Settimi, D. Caporale, P. Kryczka, M. Ferrati, and L. Pallottino, "Motion primitive based random planning for loco-manipulation tasks," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2016, pp. 1059–1066.

[18] S. Hart, P. Dinh, and K. A. Hambuchen, "Affordance templates for shared robot control," in *2014 AAAI Fall Symposium Series*, 2014.

[19] K. Bouyarmane and A. Kheddar, "Humanoid robot locomotion and manipulation step planning," *Advanced Robotics*, vol. 26, no. 10, pp. 1099–1126, 2012.

[20] ——, "Non-decoupled locomotion and manipulation planning for low-dimensional systems," *Journal of Intelligent & Robotic Systems*, vol. 91, no. 3-4, pp. 377–401, 2018.

[21] H. Dai, A. Valenzuela, and R. Tedrake, "Whole-body motion planning with centroidal dynamics and full kinematics," in *2014 IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2014, pp. 295–302.

[22] J. Englsberger, C. Ott, M. A. Roa, A. Albu-Schäffer, and G. Hirzinger, "Bipedal walking control based on capture point dynamics," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 4420–4427.

[23] D. Kim, S. J. Jorgensen, H. Hwang, and L. Sentis, "Control scheme and uncertainty considerations for dynamic balancing of passive-ankled bipeds and full humanoids," in *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2018, pp. 1–9.

[24] B. Ponton, A. Herzog, A. Del Prete, S. Schaal, and L. Righetti, "On time optimization of centroidal momentum dynamics," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1–7.

[25] F. Zacharias, C. Borst, and G. Hirzinger, "Capturing robot workspace structure: representing robot capabilities," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2007, pp. 3229–3236.

[26] K. Shoemake, "Animating rotation with quaternion curves," in *ACM SIGGRAPH computer graphics*, vol. 19, no. 3. ACM, 1985, pp. 245–254.

[27] M.-J. Kim, M.-S. Kim, and S. Y. Shin, "A general construction scheme for unit quaternion curves with simple high order derivatives," in *SIGGRAPH*, vol. 95, 1995, pp. 369–376.

[28] R. Featherstone and O. Khatib, "Load independence of the dynamically consistent inverse of the jacobian matrix," *The International Journal of Robotics Research*, vol. 16, no. 2, pp. 168–170, 1997.

[29] Y.-C. Lin, B. Ponton, L. Righetti, and D. Berenson, "Efficient humanoid contact planning using learned centroidal dynamics prediction," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 5280–5286.

[30] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[31] F. Chollet, "keras," https://github.com/fchollet/keras, 2015.

[32] R. Ebendt and R. Drechsler, "Weighted a search–unifying view and application," *Artificial Intelligence*, vol. 173, no. 14, pp. 1310–1342, 2009.

[33] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Prentice-Hall, 1995.

[34] R. A. Valenzano, N. R. Sturtevant, J. Schaeffer, and F. Xie, "A comparison of knowledge-based gbfs enhancements and knowledge-free exploration," in *Twenty-Fourth International Conference on Automated Planning and Scheduling*, 2014.

[35] D. Hershberger, D. Gossow, and J. Faust, "Rviz, 3d visualization tool for ros," *URL: http://wiki. ros. org/rviz*, 2019.

[36] Y.-C. Lin, L. Righetti, and D. Berenson, "Robust humanoid contact planning with learned zero-and one-step capturability prediction," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2451–2458, 2020.

[37] S. J. Jorgensen, J. Holley, F. Mathis, J. S. Mehling, and L. Sentis, "Thermal recovery of multi-limbed robots with electric actuators," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1077–1084, 2019.