Generating Gaits for Simultaneous Locomotion and Manipulation

Julian Whitman, Shuang Su, Stelian Coros, Alex Ansari, and Howie Choset

Abstract— **Modular robots can be rapidly reconfigured into customized articulated legged morphologies capable of mobile manipulation and inspection. However, current gait generation methods do not keep pace with the speed of physical reconfiguration. This work focuses on quickly creating gaits for modular legged robots. We build on a recent method that uses trajectory optimization to design quasi-static gaits given only robot geometry and foot contact patterns. We develop methods to automatically generate contact patterns for new gaits and transitions between them. We show the utility of these methods applied to robots with many limbs, such that limbs can be fluidly reassigned to locomotion, manipulation, or inspection tasks, or to adapt gaits to hardware failures online. We demonstrate gait and transition generation with our modular hexapod and dodecapod robots. The robots switch between gaits that use all limbs for locomotion and those that leave some limbs free to pick up objects or position a camera.**

I. Introduction

A number of researchers work on modular robots that can be reconfigured to support task needs [1], [2], [3], [4]. For instance, our own hardware [5] can be assembled into highly-articulated platforms capable of legged locomotion, mobile manipulation, and inspection [6]. While modular hardware allows us to rapidly prototype new robot morphologies, planning for locomotion and manipulation may still require significant time and computational resources. To make these systems practical, new control methods are needed that can keep pace with physical reconfiguration.

To address this challenge in the context of robot mobility, [7] presents a trajectory optimization based approach to gait design that can quickly synthesize optimal quasi-static gaits given only a robot's geometry and a desired footfall pattern. While this approach can create a wide variety of gaits for legged robots, it requires that footfall patterns be hand-crafted for each gait, and no methods have yet been developed to transition between gaits designed in this fashion.

The set of behaviors that a robot will need may not be known before deployment. For instance, a robot may need to fluidly change the role of its limbs from locomotion to manipulation or inspection. Modules and limbs may also fail, requiring the robot to adapt its gait. Rather than precomputing all possible gaits for such situations, we create new gaits online to enable rapid deployment minutes after reconfiguration. Building on [7], this work develops means to quickly and automatically synthesize footfall patterns i) to enact transitions between two known gaits and ii) to generate new candidate gaits that use a desired subset of the robot's limbs, so that remaining limbs may be used for peripheral tasks or to compensate for failures. See Fig. 1 for a depiction of the gait transition process.

The following, Sec. II provides relevant background material, including an overview of the gait design algorithm from [7]. Section III describes our approach to transition and gait generation. We demonstrate these methods on locomotively redundant morphologies, i.e., those with more limbs than necessary for quasi-static locomotion. Section IV presents hardware experiments that demonstrate our modular 18-DoF hexapod switching between hexapedal and quadrupedal gaits in order to carry a package. Similarly, we demonstrate a 12-limbed 36-DoF robot that switches gaits to locomote while simultaneously picking up objects and positioning a camera. Section IV-C describes experiments comparing our optimized gait transitions to an interpolation transition developed as a simple alternative. Section V discusses limitations of our method and directions for future work.

II. Background

A. Gait generation

There are several alternative methods used to perform gait design. A common approach is to use a central pattern generator (CPG), a system of coupled oscillators that generates rhythmic open-loop joint trajectories [8]. A CPG is created by assigning oscillator-joint pairings, then assigning parameters to the network including connection weights, oscillator frequencies, amplitudes, and offsets. These methods rely on computationally expensive global stochastic searches to find effective network parameters, for instance, requiring at least 15 minutes for robots composed of 12 or fewer modules [2], [9], [10]. With these methods, if the robot is reconfigured, expensive computation is required to modify gaits. [11] generated a low dimensional map of behaviors which could then be used to quickly regenerate gaits online after the robot was damaged, but required two weeks of precomputation for each robot design.

Other techniques have been developed to create gaits, often for narrow classes of robot morphologies. Geometric mechanics-based approaches have been used to discover gaits for snake-like robots [12], [13]. Nelder-mead optimization has been used to learn leg swing timing for a six degree-of-freedom hexapod [14]. These methods require a designer to specify motion parameterizations which are specific to one morphology, and so would not be well-suited to rapid modular reconfiguration.

All authors are at Carnegie Mellon University, 5000 Forbes Ave., Pittsburgh, PA 15213, USA. E-mail: jwhitman@cmu.edu

This work was supported by NASA Space Technology Research Fellowship NNX16AM81H and by the Robotics Collaborative Technology Alliance program.

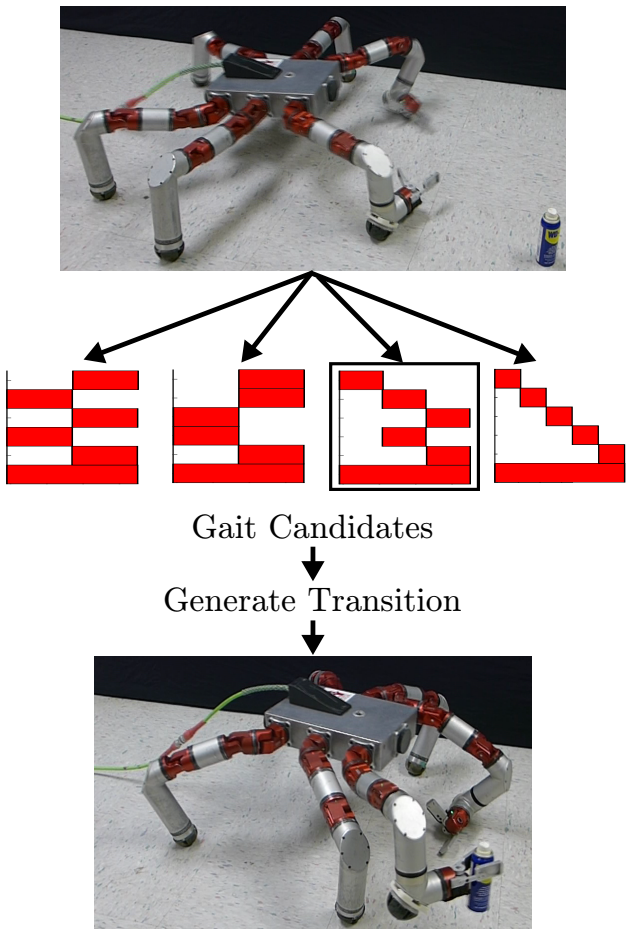

Fig. 1: We demonstrate means for a modular robot to create gaits and gait transitions, in particular, to reassign limbs from locomotion to peripheral tasks. In the above figure, the front right limb is reassigned and the robot switches from a hexapedal to a pentapedal gait. A set of candidate footfall patterns is created, and one is selected to create a gait (see Sec. III-B). Next, a transition phase to switch into the new gait is generated and executed (see Sec. III-A). Last, the new gait is executed, and the reassigned limb is freed to pick up an object while the robot locomotes.

B. Gait generation with trajectory optimization

In contrast the gait design methods mentioned previously, this work builds on a relatively recent trajectory optimization-based gait design approach from [7], since it was shown to efficiently create gaits for a variety of legged robot morphologies. In this approach, a gait for a robot with n_J joints is specified by a motion plan $\boldsymbol{P}$ consisting of a series of m discrete points in the configuration space. Each row of $\boldsymbol{P}$ is the trajectory of one joint over the cycle period, and each column is the robot configuration at time step i, $\vec{q}_i \in \mathbb{S}^{n_J}$, $\boldsymbol{P} = [\vec{q}_1 \dots \vec{q}_{m-1}, \vec{q}_m]$. Gaits are cyclical so $\vec{q}_1 = \vec{q}_m$. To generate continuous motions, at each time, t, the columns of $\boldsymbol{P}$ are interpolated with a cubic spline interpolation function to find joint angles $\vec{q}(t)$,

$$\vec{q}(t) = \text{interp}(\boldsymbol{P}, t) \tag{1}$$

The objective function for the trajectory optimization problem includes motion smoothness, and achieving user inputs including desired robot velocity, body height, body orientation, and foot step height. A sequential quadratic programming solver uses the objective analytical gradients and Hessians to concurrently generate trajectories for the robot's feet, center of mass, and full-body poses using a simplified inverted pendulum dynamics model. Constraints maintain the center of pressure within the *support polygon*, the convex hull of foot contacts, at each time step. Additional constraints enforce a no-slip condition at contacts and bound the angular velocity of joints. The resulting gaits can then be verified in a full rigid body dynamics simulation. Initial trajectory seeds are formed from a *footfall pattern* which encodes the stance or swing phases for a walking sequence, that was hand-crafted and input by a user in [7]. The footfall pattern for a robot with n_F feet over m time steps is represented by a matrix of binary contact flags $\boldsymbol{C} \in \{0,1\}^{n_F \times m}$. The contact state of each foot at each time step is either 0 for swing phase or 1 for stance phase.

The optimization process converges quickly– for instance, with 20 time steps it requires under one minute of computation time to complete for our twelve-limbed robot with 36 joints. Because of its efficiency and flexibility, we adopt this technique to create individual motions for our modular robots.

C. Transitions between gaits

One of the benefits of CPGs is that transitions between gaits can be enacted by varying the network parameters and allowing the system to stabilize to a new limit cycle [15], [16]. Similarly, transitions between gaits parameterized by step timing or step size can be implemented by varying those parameters [17], [18], [19]. These approaches rely on low-dimensional motion parameterizations selected beforehand, and so may not support rapid reconfiguration. Further, they do not apply to gaits designed with the trajectory optimization framework of [7].

III. Footfall Pattern Generation

The motion generation algorithm of [7] requires footfall patterns as an input. Hence, to synthesize new gaits that allow reassigning limb roles, new footfall patterns must be generated in which failed limbs, or limbs needed for peripheral tasks, are *nonlocomotive*, i.e. in swing phase over the full gait cycle. The remainder limbs are *locomotive*, i.e. with a stance phase within the current footfall pattern. All possible footfall patterns and corresponding motions could be precomputed offline for a given robot. However, there are a combinatorial number of footfall patterns for any fixed number of time steps. Rather than precomputing all possible motion plans for a robot, we make new motion plans online as needed to speed up deployment after reconfiguration. To create transitions between two known gaits, we pose the

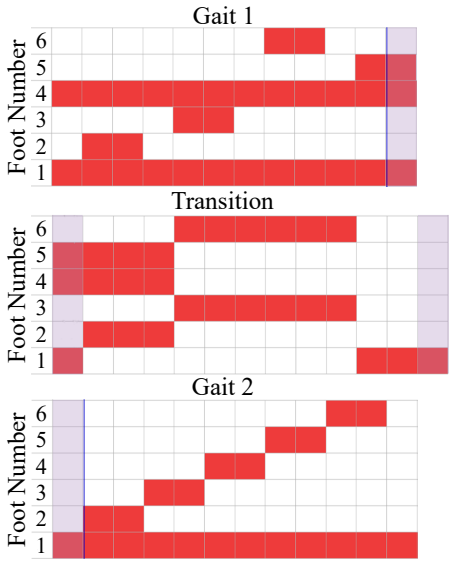

Fig. 2: Hexapod footfall patterns for a quadrupedal gait (top), pentapedal gait (bottom), and transition between them (middle). The contact state of each discrete time step is represented by a box. Red boxes represent the swing phase and white represent stance. The pattern is interpreted from left to right as the motion progresses. When limb 4 is reassigned, gait 2 is generated, then the transition is generated to connect the gaits. Shaded regions indicate boundary constraints in the transition footfall pattern generator.

footfall pattern generation as an integer program. Then we present a method to create new gaits when limbs must be reassigned.

A. *Transition generation*

This work contributes a new method of generating gait transitions by posing footfall pattern creation as an integer quadratically constrained program (IQCP). Given two known gaits and their footfall patterns, we propose a method to generate quasi-statically stable transitions in a two step process. First an IQCP is solved for a footfall pattern $\boldsymbol{C}$, and second, this footfall pattern is used as the initial seed in the trajectory optimization of [7].

While many footfall patterns could be used to create transitions, our goal is to generate motions that appear smooth and deliberate. The objective function of the integer program is therefore to minimize the number of changes between swing and stance states. We use a variable y_j to denote whether any foot switches between swing and stance at each time step j.

Then for a robot with n_F feet, we minimize the number of such time steps over the number of total time steps m,

$$\underset{C\in\{0,1\}^{(n_F\times m)}}{\text{minimize}} \sum_{j=1}^{m} y_j \qquad (2)$$
$$\text{subject to } \mathbf{g}(\boldsymbol{C}) \leq 0$$

The integer-domain minimization problem is subject to a vector of constraints $\vec{g}(\boldsymbol{C})$ which effect the transition footfall pattern. While these constraints are not derived directly from first principles, they encode requirements for quasi-static locomotion and contain strategies that help to make the initial seed in the motion plan generation feasible. See the Appendix for a full formulation of the IQCP constraints and variables.

- Constraint set $\vec{g_1}(\boldsymbol{C})$ is a limit on the number of changes between swing and stance phases for an individual foot. Over a transition period, each foot is allowed to change from swing to stance or vice versa at most twice.
- $\vec{g_2}(\boldsymbol{C})$ requires each swing phase to last more than one time step.
- $\vec{g_3}(\boldsymbol{C})$ prevents stance phases of one time step in length, and $\vec{g_4}(\boldsymbol{C})$ prevents stance phases of two time steps in length. These two heuristic constraint sets ensure that each stance phase lasts for at least three time steps, which we found to empirically result in more "natural" appearing behavior.
- Constraint set $\vec{g_5}(\boldsymbol{C})$ sets an upper limit to the total time in the transition phase each foot may spend in stance phase.
- To ensure that the feet share walking workload, $\vec{g_6}(\boldsymbol{C})$ and $\vec{g_7}(\boldsymbol{C})$ require that the difference in time spent in swing phase between any two feet is no more than two time steps.
- $\vec{g_8}(\boldsymbol{C})$ requires that at least three limbs be in stance at each time step, which helps allow stance stability.
- Not all combinations of stance feet configurations are equally desirable from a stability standpoint. For example, solutions that satisfy $\vec{g_8}$ may include patterns where all feet on the left side of a hexapedal robot are in swing at once, while the others are in stance. Constraint sets $\vec{g_9}(\boldsymbol{C})$ and $\vec{g_{10}}(\boldsymbol{C})$ ensure that at least one limb on each side is in stance at any moment in time.

As transition points, we select for simplicity the final point in the first gait and the starting point in the second gait. These are used both as constraints for the footfall pattern and for the trajectory optimization. The constrained minimization is solved with a Gurobi optimizer, a state-of-the-art integer solver [20]. The solver outputs transition footfall patterns in a few seconds. For instance, the transition pattern shown in Figure 2, in which a transition for six limbs over 11 time steps, $\boldsymbol{C} \in \{0,1\}^{6\times 11}$, was generated in 4.5 s with a 2.5 GHz Intel Core i7 processor. Similarly, the footfall pattern used for the twelve-limbed robot over 18 time steps, $\boldsymbol{C} \in \{0,1\}^{6\times 18}$, was generated in 6 s.

As a simple alternative transition method, we directly interpolate between two gaits. This is a straightforward, common method of transitioning between compatible gaits [19]. For our gaits, we interpolate between two matrices of joint angles over a gait cycle. The initial gait $\boldsymbol{P}_a$ and final gait $\boldsymbol{P}_b$ produce joint angles over time $\vec{q}_a(t)$ and $\vec{q}_b(t)$ as in (1). Over the course of one cycle period T, the joint angles for the interpolated transition at time t are:

$$\vec{q}(t) = \vec{q}_a(1 - \frac{t}{T}) + \vec{q}_b\frac{t}{T} \qquad (3)$$

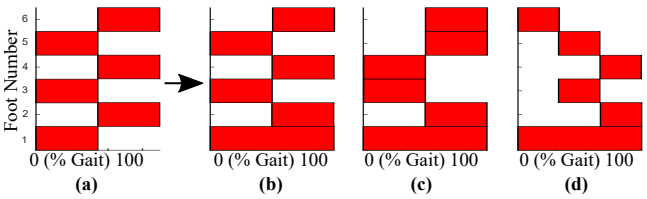

Fig. 3: Candidate footfall patterns for hexapod gait generation when one limb must become nonlocomotive. To avoid testing every possible pattern, a small set of patterns similar to the original are tested. Red regions represent swing states, and white regions are stance states for each limb. (a) depicts the original alternating tripod gait. After the reassignment of limb 1, candidate patterns with limb 1 nonlocomotive are tested: (b) the original pattern with limb 1 nonlocomotive, (c) a pattern in which the swing phase of 4 and 5 are switched, and (d) a pattern in which the swing phase of limbs 4 and 6 are separated.

This method is computationally simple, but as we show in Sec. IV-C, may lead to the robot falling over or moving in undesired directions.

B. Gait generation

In the procedure described in [7], footfall patterns were input to the motion generator by a user. To enable a higher level of autonomy for a rapidly assembled modular robot, we tested methods for generating a footfall pattern when a subset of limbs are reassigned to new roles. For instance, limbs may be needed for a new manipulation or inspection task, or a limb may no longer be usable due to hardware failure. We developed three footfall pattern generation methods: 1) If only one limb will be reassigned from locomotive to nonlocomotive, then we make a small modification to the current footfall pattern. 2) If more than one limb will be reassigned, then we use a pattern generation procedure similar to the IQCP described in Section III-A. If the resulting patterns from (1) or (2) do not result in feasible gaits, then 3) we create a simple wave gait for the locomotive limbs.

1) One limb changes from locomotive to nonlocomotive: If only one limb will change from a locomotive to nonlocomotive role, as might occur in the case of a failed limb, then for many-limbed robots a small variation may be sufficient to regenerate an effective gait, so we create a footfall pattern similar to the original pattern. A set of candidate gait footfall patterns are derived from the original pattern, modified to make the failed limb nonlocomotive, and differ by two types of operations. In the first type of operation, the swing phase of one limb which overlaps with another limb's swing phase is separated into its own step. This type of operation tends to evolve the pattern towards a wave gait pattern where each locomotive limb swings separately. Wave gaits have a higher stability margin [21], but tend to be slower, than gaits with longer swing phases like alternating tripod. In the second type of pattern modification operation, the swing phases of two locomotive limbs are switched. This allows gaits created from modified patterns to form different support polygon shapes. See Figure 3 for an example of candidate patterns.

These two operations on the original footfall pattern result in a set of candidate patterns, one for each unique switched swing phase or separated swing phase. To select which candidate pattern should be used in the gait trajectory optimization, the patterns are ranked with a static stability heuristic. At each time step in a candidate pattern, the distance from the center of mass to the centroid of the support polygon is computed. Each foot placement during this calculation is the average location of the foot in stance phase during the original gait. The distances across time steps are then summed. The best candidate patterns by this metric are used to generate gaits. Depending on the computational resources available, multiple gaits can be generated from these candidate patterns, potentially in parallel, and the final gait selected based on the cost at convergence. A gait with a lower cost better fulfills optimizer objectives, such as having a walking speed closer to the desired speed.

2) Other limb reassignments: The above method only applies when one limb is reassigned from a locomotive to a nonlocomotive role. When this is not the case, an entirely new footfall pattern can be generated using an IQCP. The procedure for this generation is the same as that of transition generation, with the exception of altered boundary constraints. In the transition generation, the boundary constraints are set to match the transition points in the first and second gaits. Gait footfall patterns are cyclical, so these constraints are replaced with a periodicity requirement. This algorithm, unlike the previous, will only generate one contact pattern.

3) Fallback gait option: We observe that occasionally, the motion optimization process of [7] fails to find satisfactory motion plans when initialized with footfall patterns generated via the two methods above. In this case, a wave gait footfall pattern can be used. The locomotive limbs are each assigned a small swing phase such that only one locomotive limb is in swing phase at each time step.

IV. Gait creation and transition experiments

A. Robot hardware

To test our methods, we conducted experiments in which we reconfigured our modules into two different legged robots. The first, a hexapod, was composed of 20 modular actuators [5], three for each of its six limbs and one for each of its two grippers. The second, a dodecapod, was composed of 38 modular actuators, three for each of its 12 limbs and one for each of its two grippers. A camera was mounted on the end of one leg for inspection tasks. Each actuated module contains a low-level processor which tracks desired set points. Nonlocomotive limbs were controlled with a gamepad joystick and transitions between gaits were triggered via gamepad buttons.

B. Limb role reassignment demonstrations

We demonstrated our approach to gait and transition generation with a hexapod and a dodecapod robot. We emphasize the utility of our method for redundant locomotors, like our

Hexapod initial gait $\rightarrow$ final gait	Direct interpolation velocity (cm/s)	Optimized transition velocity (cm/s)
$4 \rightarrow 5$	6.4	6.4
$4 \rightarrow 6$	6.4	6.2
$5 \rightarrow 4$	0.08	5.9
$5 \rightarrow 6$	5.2	5.9
$6 \rightarrow 4$	–4.0	5.2
$6 \rightarrow 5$	4.8	6.0

TABLE I: A comparison of the simulated average forward velocity during transitions on a hexapod robot. The "direct interpolation" is created by linearly interpolating between the joint trajectories of two gaits. "Optimized" transitions are created using our two step optimization procedure. Gaits are labelled by the number of locomotive limbs: "6" for an alternating tripod hexapedal gait, "5" for a pentapedal gait in which one front limb is nonlocomotive, "4" for a quadrupedal gait in which the front limbs are nonlocomotive. The walking speed of the interpolated transition is in many cases much lower than the speed of both the 6.4 cm/s first and second gaits. In one case, $6 \rightarrow 4$, the interpolated transition moves the robot in the wrong direction.

robots, where limbs can be reassigned simultaneously to peripheral tasks while still leaving enough limbs for quasi-static locomotion. In each demonstration, the robot begins with all limbs in locomotive roles. We choose a subset of limbs to be reassigned, and create a new gait. We then create a transition that maintains robot heading, orientation, and speed. These examples show different sets of limbs used to pick up and carry objects or to position a camera while the robots locomote. Here we computed motion plans offline, but because each motion plan takes under one minute to create, they could be computed online in response to new tasks. To emulate limb failures, between one and four limbs on the dodecapod were randomly reassigned as nonlocomotive. Our method was able to generate new gaits to compensate for the loss of function, with the limitation that we assume failed limbs remain in a relatively fixed position. The trajectory optimization failed when the robot geometry did not allow locomotive limbs to form a quasi-static support base, for example, when two limbs on the left side of the hexapod were made nonlocomotive. Other than these cases, we found this method could generate gaits that allow limbs to be used both for peripheral tasks and locomotion, and could generate transitions between any two of those gaits. Snapshots of these behaviors are shown in Figures 1, 4, and 5. The full gait transition sequences are shown in the supplementary video.[1]

C. Comparison to interpolated transitions

In the experiments in Table I, each initial and final gait were selected from hexapod gaits demonstrated in Section IV-B, which have varying combinations of locomotive and nonlocomotive limbs. Interpolated transitions, created using Equation 3, are computationally inexpensive. But, we find that they may not achieve objectives such as maintaining walking speed and heading. The optimized transitions consistently move the robot forward, but the interpolated transitions do not.

[1]The supplementary video can be found at `https://youtu.be/xXfi55HPOfs`

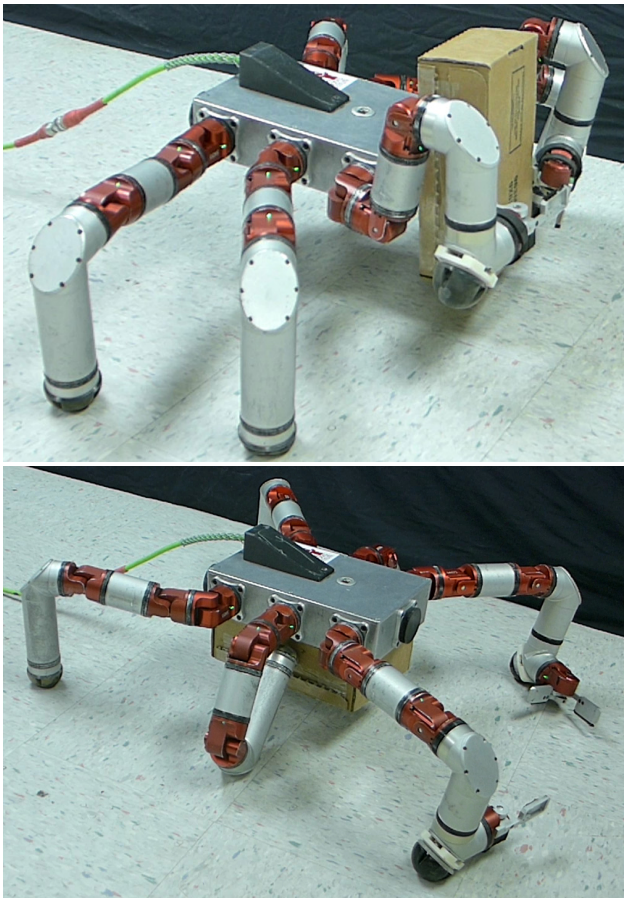

Fig. 4: A quadrupedal gait allows the use of the front limbs to grasp and carry an object (middle). A different quadrupedal gait allows the use of the middle limbs to grasp and carry an object (bottom)

Interpolated and optimized transitions were also tested in simulation for a dog-like 12-DoF quadruped. Two different wave gaits were created, both of which were quasi-statically stable and moved the quadruped forward. The interpolated transitions did not maintain stability, and the quadruped fell. An optimized transition was able to move between these gaits while maintaining stability and forward speed, further demonstrating the utility of transitions created through trajectory optimization.

V. Conclusions

Robots with many limbs have high redundancy– limbs can be reassigned to other roles without preventing quasi-static locomotion. Reassigned limbs can be used for manipulation, inspection, or left unused after limb hardware failures. We

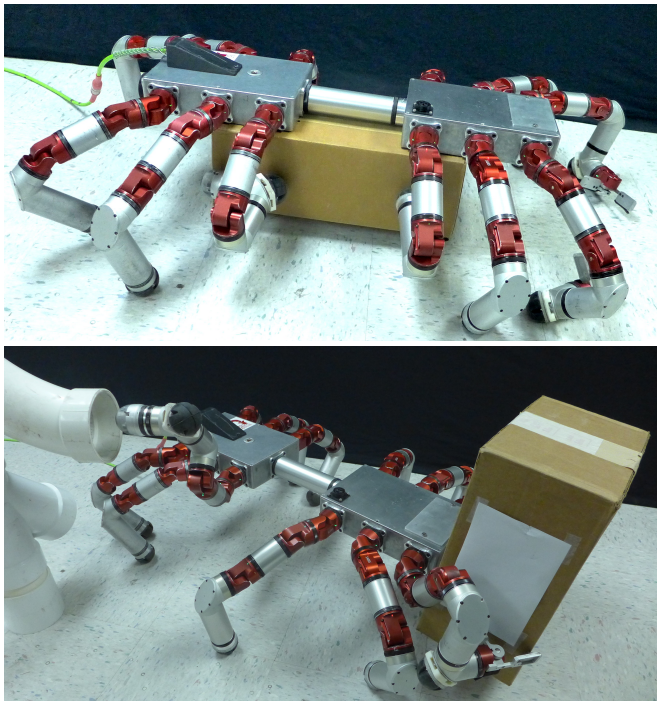

Fig. 5: Dodecapod robot using its many limbs for manipulation or inspection tasks. The four middle limbs can carry a larger box than can any two limbs alone (top). One limb is used to position a camera to inspect a pipe opening while the front of the robot carries a package (bottom).

presented methods to create quasi-static gaits and transitions, allowing us to reassign limbs on-the-fly.

Future work will extend the methods of [7] to create highly dynamic gaits. This would allow the generation of faster-moving gaits, especially for morphologies such as bipeds and tripods. Additionally, this would allow gaits to better account for the under-actuated motion of failed limbs, in contrast to our current methods that assume failed limbs remain in a fixed position. A limitation of our method is that footfall pattern creation is largely decoupled from the robot morphology and the motion plan generation. Future work will investigate methods that explicitly consider the interplay between these factors without a significant increase in computation time.

We envision these methods of gait and transition generation as part of a full modular robot deployment process. While the examples we present in this work show robots locomoting in a straight line, these methods can be used to allow legged robots to locomote in any direction. A variety of gaits with varying locomotion directions could be generated in parallel on multiple processors since each gait is independent. A high-level planner could specify the sequence of gait headings to follow a path, and build over time a library of available gaits as needed. This work assumes that the user designates which points on the robot can be considered as "feet" during planning. Future work will incorporate automatic recognition of potential contacts based on module type and placement. Also, users must currently select which legs should be reassigned, and control their motion with a joystick. Future work will focus on the development of automated tools for planning the reassignment and control of nonlocomotive limbs. With such tools in hand, a modular robot could be reconfigured, given high level task descriptions, and deployed in a matter of minutes.

APPENDIX

The integer quadratically constrained program (IQCP) we solve to generate footfall patterns requires a linear objective function to be minimized subject to linear and quadratic constraints. In this appendix, we formally describe the optimization problem described in Section III-A, which is defined in full as:

$$\begin{aligned} &\underset{C \in \{0,1\}^{(n_F \times m)},}{\text{minimize}} \quad \sum_{j=1}^{m} y_j \\ &y \in \{0,1\}^{(m-1)}, \\ &z \in \{0,\dots n_F\}^{(m-1)} \\ &\text{subject to} \\ &\mathbf{h}(C,y,z) = 0 \\ &\mathbf{g}(C,y,z) \leq 0 \end{aligned}$$

Here, $C \in \{0,1\}^{(n_F \times m)}$ denotes the set of binary variables that encode swing or stance phases for each limb $i \in \{1,\dots n_F\}$ of the robot at each time sample $j \in \{1,\dots m\}$. The set of variables $y \in \{0,1\}^{(m-1)}$ corresponds to the number of transitions between stance and swing phases, or vice versa, at each time sample, which is computed with the aid of auxiliary variables $z \in \{0,\dots n_F\}^{(m-1)}$ and constraints $\mathbf{h}(C,y,z)$. The first equality constraint, h_1, sets z_j to be equal to the number of transitions between swing and stance phases,

$$h_1 = z_j - \sum_{i=1}^{n_F} (C_{i,j} - C_{i,j+1})^2 = 0, \quad \forall j \in \{1,\dots m-1\}$$

The second equality constraint h_2 sets y to be 1 for every time step that exhibits at least one transition between phases, and 0 otherwise,

$$h_2 = (y_j - 1) z_j = 0, \quad \forall j \in \{1,\dots m-1\}$$

While the number of transitions between swing and stance phases is used as a regularizer, the constraints $\mathbf{g}(C)$ that guide the emergence of transition foot fall patterns, as described in Section III-A, are formally defined here.

Constraint set g_1 is a limit on the number of changes between swing and stance phases for an individual foot. Over a transition period, each foot is allowed to change from swing to stance, or vice versa, at most twice,

$$g_1 = \sum_{j=1}^{m-1} (C_{i,j} - C_{i,j+1})^2 - 2 \leq 0, \quad \forall i \in \{1,\dots n_F\}$$

g_2 requires each swing to phase last more than one time step,

$$g_2 = C_{i,j} - C_{i,j+1} + C_{i,j+2} - 1 \leq 0,$$
$$\forall i \in \{1, \dots n_F\}, j \in \{1, \dots m-2\}$$

g_3 prevents stance phases of one time step in length,

$$g_3 = -C_{i,j} + C_{i,j+1} - C_{i,j+2} \leq 0,$$
$$\forall i \in \{1, \dots n_F\}, j \in \{1, \dots m-2\}$$

and g_4 prevents stance phases of two time steps in length,

$$g_4 = -C_{i,j} + C_{i,j+1} + C_{i,j+2} - C_{i,j+3} - 1 \leq 0,$$
$$\forall i \in \{1, \dots n_F\}, j \in \{1, \dots m-3\}$$

These two heuristic constraint sets ensure that each stance phase lasts for at least three time steps, which we found to empirically result in more "natural" appearing behavior. g_5 sets an upper limit to the total length of the transition phase each foot may spend in stance phase, forcing each limb to be in swing for at least two time steps,

$$g_5 = \sum_{j=1}^{m} C_{i,j} - m + 2 \leq 0, \quad \forall i \in \{1, \dots n_F\}$$

To ensure that the feet share walking workload, g_6 and g_7 requires that the difference in time spent in swing phase between any two feet is no more than two time steps,

$$g_6 = \sum_{j=1}^{m} C_{i_1,j} - \sum_{j=1}^{m} C_{i_2,j} - 2 \leq 0,$$
$$g_7 = -\sum_{j=1}^{m} C_{i_1,j} + \sum_{j=1}^{m} C_{i_2,j} - 2 \leq 0,$$
$$\forall i_1 \in \{1, \dots n_F\}, i_2 \in \{1, \dots n_F\}, i_1 \neq i_2$$

g_8 requires at least three limbs in stance at each time step,

$$g_8 = -\sum_{i=1}^{n_F} C_{i,j} + n_F - 3 \leq 0, \quad \forall j \in \{1, \dots m\}$$

This constraint helps ensure stance stability. However, not all combinations of stance feet configurations are equally desirable from a stability standpoint. For example, solutions that satisfy g_8 would include all feet on the left side of a hexapedal robot being in swing, while the others are in stance. Constraints g_9 and g_{10} ask that at least one leg on each side is in stance at any moment in time.

$$g_9 = -\sum_{i=1}^{n_F/2} C_{i,j} + 1 \leq 0, \quad \forall j \in \{1, \dots m\}$$

$$g_{10} = -\sum_{i=n_F/2+1}^{n_F} C_{i,j} + 1 \leq 0, \quad \forall j \in \{1, \dots m\}$$

REFERENCES

[1] T. Tosun, G. Jing, H. Kress-Gazit, and M. Yim, "Computer-aided compositional design and verification for modular robots," in *International Symposium on Robotics Research*. Citeseer, 2015.

[2] S. Bonardi, M. Vespignani, R. Moeckel, J. V. den Kieboom, S. Pouya, A. Sproewitz, and A. Ijspeert, "Automatic generation of reduced CPG control networks for locomotion of arbitrary modular robot structures," in *Proceedings of Robotics: Science and Systems*, Berkeley, USA, July.

[3] M. Yim, W.-M. Shen, B. Salemi, D. Rus, M. Moll, H. Lipson, E. Klavins, and G. S. Chirikjian, "Modular self-reconfigurable robot systems [grand challenges of robotics]," *IEEE Robotics & Automation Magazine*, vol. 14, no. 1, pp. 43–52, 2007.

[4] K. Stoy, D. Brandt, and D. J. Christensen, *Self-reconfigurable robots: an introduction*. MIT Press, 2010.

[5] D. Rollinson, Y. Bilgen, B. Brown, F. Enner, S. Ford, C. Layton, J. Rembisz, M. Schwerin, A. Willig, P. Velagapudi, and H. Choset, "Design and architecture of a series elastic snake robot," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2014, pp. 4630–4636.

[6] A. Ansari, J. Whitman, B. Saund, and H. Choset, "Modular platforms for advanced inspection, locomotion, and manipulation," in *Waste Management Conference, (In Press)*, 2017.

[7] V. Megaro, B. Thomaszewski, M. Nitti, O. Hilliges, M. Gross, and S. Coros, "Interactive design of 3d-printable robotic creatures," *ACM Transactions on Graphics (TOG)*, vol. 34, no. 6, p. 216, 2015.

[8] A. J. Ijspeert, "Central pattern generators for locomotion control in animals and robots: a review," *Neural networks*, vol. 21, no. 4, pp. 642–653, 2008.

[9] D. J. Christensen, J. C. Larsen, and K. Stoy, "Fault-tolerant gait learning and morphology optimization of a polymorphic walking robot," *Evolving Systems*, vol. 5, no. 1, pp. 21–32, 2014.

[10] V. Vonásek, S. Neumann, D. Oertel, and H. Wörn, "Online motion planning for failure recovery of modular robotic systems," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 1905–1910.

[11] A. Cully, J. Clune, D. Tarapore, and J.-B. Mouret, "Robots that can adapt like animals," *Nature*, vol. 521, no. 7553, pp. 503–507, 2015.

[12] C. Gong, D. I. Goldman, and H. Choset, "Simplifying gait design via shape basis optimization," in *Proceedings of Robotics: Science and Systems*, Ann Arbor, Michigan, June 2016.

[13] J. Dai, H. Faraji, C. Gong, R. L. Hatton, D. I. Goldman, and H. Choset, "Geometric swimming on a granular surface," in *Proceedings of Robotics: Science and Systems*, Ann Arbor, Michigan, June 2016.

[14] J. D. Weingarten, G. A. Lopes, M. Buehler, R. E. Groff, and D. E. Koditschek, "Automated gait adaptation for legged robots," in *International Conference on Robotics and Automation (ICRA)*, vol. 3. IEEE, 2004, pp. 2153–2158.

[15] A. J. Ijspeert, A. Crespi, D. Ryczko, and J.-M. Cabelguen, "From swimming to walking with a salamander robot driven by a spinal cord model," *Science*, vol. 315, no. 5817, pp. 1416–1420, 2007.

[16] W. Chen, G. Ren, J. Zhang, and J. Wang, "Smooth transition between different gaits of a hexapod robot via a central pattern generators algorithm," *Journal of Intelligent & Robotic Systems*, vol. 67, no. 3-4, pp. 255–270, 2012.

[17] G. C. Haynes and A. A. Rizzi, "Gaits and gait transitions for legged robots," in *IEEE International Conference on Robotics and Automation*. ICRA, 2006, pp. 105–121.

[18] M. Travers, A. Ansari, and H. Choset, "A dynamical systems approach to obstacle navigation for a series-elastic hexapod robot," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, Dec 2016, pp. 5152–5157.

[19] C. Gehring, S. Coros, M. Hutter, M. Bloesch, M. A. Hoepflinger, and R. Siegwart, "Control of dynamic gaits for a quadrupedal robot," in *Robotics and automation (ICRA), 2013 IEEE international conference on*. IEEE, 2013, pp. 3287–3292.

[20] Gurobi optimizer reference manual. Accessed 2017-02-28. [Online]. Available: http://www.gurobi.com/resources/getting-started/mip-basics

[21] S.-M. Song and K. J. Waldron, "An analytical approach for gait study and its applications on wave gaits," *The International Journal of Robotics Research*, vol. 6, no. 2, pp. 60–71, 1987.