2014 American Control Conference (ACC)
June 4-6, 2014. Portland, Oregon, USA

Planning End Effector Trajectories for a Serially Linked, Floating-base Robot with Changing Support Polygon

Ellen A. Cappo, Howie Choset

Abstract—**We present a framework for the planning of end effector trajectories for serially linked robots with a non-fixed, also known as floating, support base. The FBSR (Floating Base, Serial Reach) planning algorithm extends the capabilities of serially linked, floating-base robots by planning to maintain stability while transferring modules from a weight-bearing role to the active, manipulator-like portion of the robot as needed to extend the robot's reach. During planning, we employ several methods such as random restarts and dynamic weighting of the optimization cost function sub-goals to reliably provide stable solution trajectories. We experimentally validate this algorithm on a 16 degree of freedom snake robot, and demonstrate the successful execution of generated plans.**

I. Introduction and Related Work

Hyper-redundant, serially linked modular robots—commonly referred to as snake robots—have long been studied for their ability to maneuver through small spaces and traverse uneven terrain. Their locomotive capabilities make them well suited for applications such as search and rescue, where traversing through rubble and debris is critical, and for inspection tasks, especially of piping systems such as sewers and power plants.

Due to their maneuverability, much of the published research for snake-like robots has been on forms of locomotion, especially gaited locomotion [1]–[3]. Although gaits can be adapted to work over a variety of terrains [4] and gait parameters can even be optimized for speed or energy efficiency [5], unknown obstacles larger than a few link-lengths are difficult to overcome using pre-scripted motion patterns [1].

For applications where pre-scripted motions of snake robots fail, we look at deliberately planning trajectories for these high degree of freedom systems that can incorporate environment knowledge and satisfy a user-directed goal. One such difficult but useful behavior for snake robots is raising modules off the ground to avoid obstacles, cross gaps, and observe or inspect environment features. In this paper, we present a planning framework to lift modules off the ground and move the end effector towards a given goal. This planning method explicitly ensures the stability of the system as it raises links, optimizing between reaching the desired position or orientation and maintaining stability.

While balancing has not generally been a large concern for snake robots due to the low profile of most snake locomotion methods, balance is critical for many other floating-base robots. Unlike industrial manipulators which commonly have a link fixed relative to an inertial frame, the reference frame of a floating-base robot is mobile relative to the inertial frame of the world [6]. In the bipedal robot literature, the zero-moment point (ZMP) [7] is a standard measure for quantifying the stability of a balancing system. Additionally, the ZMP criteria for changing the geometry of a support polygon, where the support polygon is the convex hull of the robot's ground contact points, to extend a robot's reach has been shown for bipedal robots [8]. Due to the difference in configuration between a biped and a snake-robot, however, we constrain ourselves to a more static approach; humanoids can afford to step while snake robots cannot.

In the snake-robot literature, methods that use the mechanics of serial kinematics to perform manipulation-like tasks can be roughly divided into two groups: methods which manipulate the head of the snake but fix the roles of links as either ground-contacting or manipulating, and methods which transfer links from a prone to a vertical position but do not allow for arbitrary goal configurations.

Of the former methods, Yamakita, Hashimoto, and Yamada [9] and most recently, Tanaka and Matsuno [10], consider tracking a trajectory with the raised head of a snake-like robot. These methods iteratively optimize a cost function at each time step to determine joint velocities. Although Yamakita does not consider system stability, Tanaka and Matsuno explicitly optimize over the position of the ZMP in relation to a support polygon in their cost function. In both works, however, the roles of the links are fixed, and modules designated as base modules cannot transfer to the raised portion to extend the system's reach.

An early example of iteratively raising modules of a serial-linked robot off the ground was given by Nilsson [11]. Nilsson's "free-climbing" algorithm profiles the torque required by a joint to lift increasing numbers of modules into the air. While allowing for iterative application of lifting increasing numbers of modules, Nilsson explicitly neglects stability measures. Ye, Ma, Li, and Wang look at using a modified serpenoid curve to lift modules of a planar snake [12]. Ye shows that the proposed curve does not compromise stability, but like Nilsson's method, these scripted motions do not incorporate knowledge of the environment or allow for the planned motion to change to better reach a target configuration.

The limitation of scripted motions to adapt to different environments coupled with the lack of reach-extension for current optimization based approaches shows the need for a planning framework that can provide the coupled behaviors. In this work, we look to extend the capabilities of snake robots and other serially linked, floating base platforms by

E. A. Cappo, and H. Choset are with the Robotics Institute at Carnegie Mellon University, Pittsburgh, PA 15213 USA (e-mail: eacappo@cmu.edu, choset@cs.cmu.edu).

using planning techniques and stability measures adapted from the manipulator and bipedal robot communities to enable planning for, and execution of, balance critical maneuvers.

II. Problem Statement and Approach

We consider the problem of planning joint trajectories for a floating base, serially linked robot. The planned joint trajectories must simultaneously satisfy the stability requirements of the system and move the end-effector towards a goal in the workspace. In this section, we present a high level overview of the Floating Base, Serial Reach planning algoirthm (FBSR); methodology and implementation specific details of the FBSR planner are described in the following section.

FBSR moves the robot towards a user defined target position or orientation by incrementally adding modules to the manipulator portion of the robot until the robot is within a specified ending criteria. The ending criteria may be as simple as a limit on the number of modules transitioned to the active section, or a check to ensure that the robot is within an error margin of the target pose. FBSR returns a final trajectory formed by concatentating intermidate trajectories; each itermediate trajectory segment alternates between moving the end-effector towards the goal, or shifting the active portion of the robot into a stable position in preparation for lifting additional modules from the ground.

We initialize the robot by designating the end-effector, i.e. the head module, and a chosen number of adjacent modules as active modules. These modules make up the manipulator portion of the robot. The remaining modules form the base portion, and the convex hull of their ground contact points forms the robot's support polygon. An example of active and base portions is illustrated in Figure 1. The robot must begin in a stable position, with the ZMP of the active modules inside the support polygon formed by the base modules.

We then plan a trajectory for the active portion of the robot by optimizing over a cost function. The cost function minimizes the error between the robot's position and the user defined goal and rewards greater distances between the ZMP and the edge of the support polygon formed by the base modules. While trajectories may be planned using a variety of existing methods such as RRTs or probabilistic roadmaps, we have chosen to optimize over the restricted space of cubic spline trajectories. These trajectories give enough freedom to find valid solutions, while providing tractable limits to the search space through encoding desired smoothness constraints.

After planning a trajectory for the initial active joints, the ending pose of the robot is checked against a pre-defined ending condition. If the ending condition is met or the number of planning cycles has exceeded a threshold, the planner terminates. If the robot has not met the ending criteria, the robot does not have enough modules in the manipulator portion to reach the goal and additional modules must be added to the manipulator. Before this can be done, the robot must be moved to a position that will be stable when modules are moved from the base. The ending

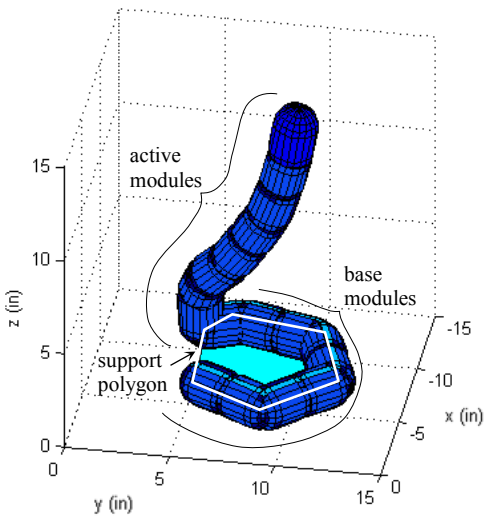

Figure 1(a)

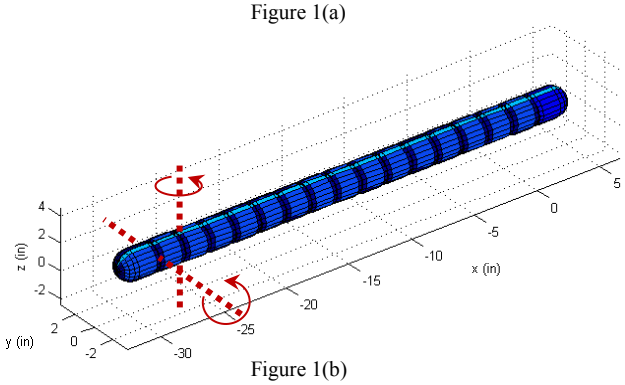

Figure 1(b)

Figure 1. (a) A serially linked, modular robot showing active and base designations as well as the support polygon. (b) Alternating axes of rotation.

configuration of the previous plan is therefore used as the starting configuration for a new plan to transfer modules from the base to the active portion to extend the manipulator's reach.

In order to stably transfer modules from a supporting role to an active role, we must examine the ZMP's position relative to both the current support polygon and the projected support polygon—the polygon that will be formed by the remaining base modules after transferring modules to the active position. The ZMP must be within the polygon formed by the intersection of the current support polygon, U, and the next planned support polygon U'. We express this as:

$$ZMP \in U \cap U' \quad (1)$$

If the ZMP is not contained in the polygon intersection, we plan trajectories for the current active portion which shifts the ZMP inside the polygon intersection by optimizing over a cost function which rewards greater distances between the ZMP and the edge of the polygon intersection. Once this has been accomplished, the modules are transitioned to the active portion and the planning loop repeats.

III. Methodology and Implementation

We have implemented the FBSR algorithm described in the preceding section on a 16 degree of freedom snake robot system. The snake robot on which we tested our algorithm is composed of 17 modules with a corresponding 16 joints,

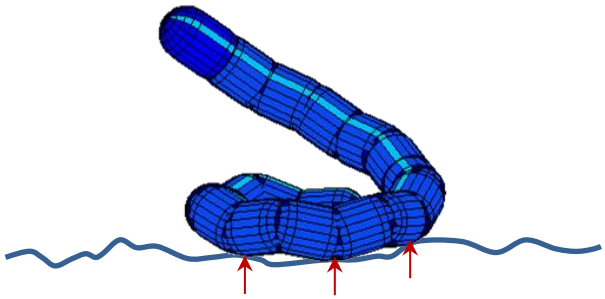

Figure 2. Exaggerated illustration of enforced ground contact points.

with the joints alternating vertical / horizontal axes of rotation along the body of the robot, as shown in Figure 1. A detailed presentation of the robot's physical parameters and architecture is described in [13].

In this section, we briefly review the details of our implementation, including stability calculations and choice of trajectory representation. We then present the cost function used during the planning and optimization, and discuss error checking and the dynamic weighting strategy.

A. *Forward Kinematics*

The chosen cost function depends implicitly upon the forward kinematics. The positions of the modules, as defined in the world frame, are used to find the support polygon, the zero moment point, and the position of the head / end-effector.

The kinematics of a serial linked, rigid body system are well understood [14]. Therefore we briefly note that the construction of the robot is as described in [13] and as illustrated in Figure 1. Modules are connected by revolute joints, and the axis of rotation of the joints alternates between the lateral and dorsal direction of the snake. We index all modules to the first, i.e. head or end-effector, link of the robot, and then describe the modules relative to the inertial frame of the world with the homogeneous transform g, shown as:

$$g_w^i = \begin{bmatrix} R_w^i & p_w^i \\ 0 & 1 \end{bmatrix}, \tag{2}$$

where g describes the rotation, R, and position, p, of a module with index i relative to w, the world reference frame.

B. *Support polygon and ZMP*

The support polygon of the robot is formed by the ground contact points of the base. Without a direct way to sense contact, there is uncertainty in the location of the contact points; due to joint offset error and ground roughness, the modules of the robot may not contact the ground evenly. To address this, we set the base joints to a slight offset value in order to force the robot to rest on the joints of every other module, as shown in Figure 2. The positions of the modules, and therefore the ground contact points, relative to the inertial frame can then be found using forward kinematics.

While the zero moment point may be calculated using the dynamics of the system [7], we here take a quasi-static approach as the robot moves slowly to accurately follow the planned trajectory. We therefore define the zero moment point of the system as the point on the ground at which the net moment of the gravitational forces equals zero in the horizontal plane, i.e., the projection of the center of gravity onto the ground plane.

C. *Choice of trajectory representation*

To provide a tractable search space, we choose to represent the trajectories of individual joints as cubic splines because of the flexibility of the method, the ensured smoothness of the generated path, and the ease of optimization over spline parameters. We place a velocity constraint on the endpoints of the trajectory and a position constraint at the start of the spline to ensure compatibility between successive trajectory segments, and we additionally specify inequality constraints representing the joint limits.

The trajectory of each active joint is represented as a cubic polynomial, of the form

$$\theta(t) = at^3 + bt^2 + ct + d \tag{3}$$

The initial position of the joint is known, so when t = 0,

$$\theta(0) = \theta_0 = d \tag{4}$$

We also require each joint to begin and end at rest, i.e., that the velocity of the joint is zero at t0 and at t_{end}.

$$\dot{\theta}(t) = 3at^2 + 2bt + c \tag{5}$$

$$\dot{\theta}(t_0) = \dot{\theta}_0 = c = 0 \tag{6}$$

$$\dot{\theta}(t_{end}) = 3at_{end}^2 + 2bt_{end} = 0 \tag{7}$$

$$a = -\frac{2bt_{end}}{3t_{end}^2} \tag{8}$$

Equation (8) defines the relationship between a and b, and d and c are determined as in Equations (4) and (6), respectively. This gives a one parameter family of functions over which to optimize. We have found that the simple position and velocity constraints as described here provide

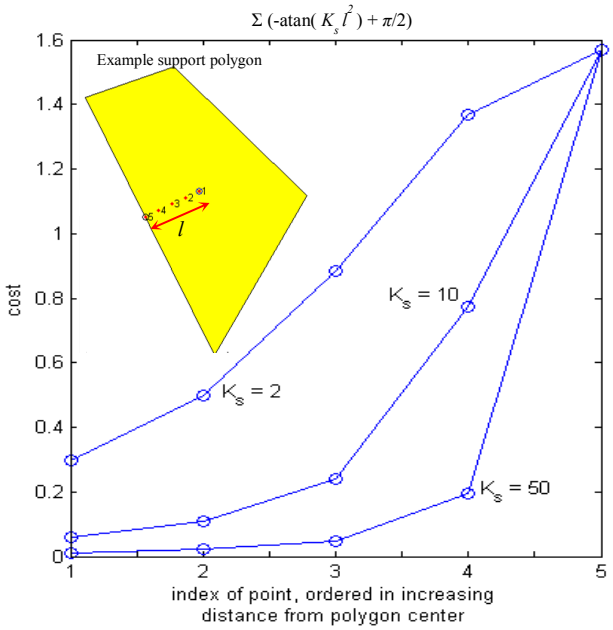

Figure 3. Example of effects of K_s on the function slope at the interior polygon edge

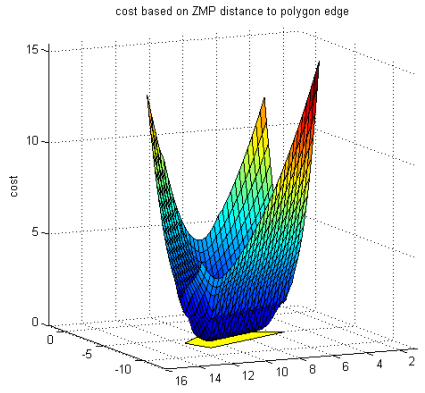

Figure 5. Illustration of C_s inside and outside the polygon. $K_s = 10$.

ample freedom for trajectory optimization. Using splines of higher order or connecting cubic splines through the use of knot points is a simple extension if further degrees of freedom are needed.

D. *Cost function*

The cost function is made up of three weighted sub-terms,

$$C = K_1 C_d + K_2 C_s + K_3 C_p \tag{9}$$

where the sub-costs C_d, C_s, and C_p are defined as follows.

C_d is the *directive* cost. The directive is the user-specified goal for the robot, such as moving to a defined position or orientation. For our application, the directive cost is a cost based on the negative z position of the head module at the end of the generated trajectory. Raising the head of the robot, i.e. maximizing the z position, is equivalent to minimizing the negative z position. The forward kinematics as described in Section A, FK: $\mathbb{R}^{16} \rightarrow SE(3)$, map the joint angles to Euclidean space, and the ending head position is found as described in Equation (11).

$$\bar{\theta}_{end} = \bar{a}(t_{end})^3 + \bar{b}(t_{end})^2 + \bar{c}(t_{end}) + \bar{d} \tag{10}$$

$$FK_{p^1}(\bar{\theta}_{end}) = \mathrm{p}^1_{\mathrm{w,t_{end}}} = \begin{bmatrix} \mathrm{x}^1_{\mathrm{w,t_{end}}} \\ \mathrm{y}^1_{\mathrm{w,t_{end}}} \\ \mathrm{z}^1_{\mathrm{w,t_{end}}} \end{bmatrix} \tag{11}$$

$$C_d = -z^1_{w,t_{end}} \tag{12}$$

We note that the directive cost does not need to be the negative height of the head; we have chosen this as the simplest function that produces a rising behavior but one could also choose to reduce the error between a desired pose, target position, or both.

C_s is the *stability* cost, and is based on the distance of the ZMP to the closest polygon edge. If our cost function consisted solely of the directive cost, the optimization yields joint trajectories that move the active portion of the snake vertically into the air. This does indeed maximize the height, but does nothing to satisfy stability requirements. We therefore add the term C_s to the cost function based on the ZMP's position in the support polygon to assist with stability optimization. C_s is based on l, the distance between the ZMP and the closest polygon edge as shown in Figure 3. We shape the cost function based on l as follows:

$$C_s(t) = \begin{cases} -\operatorname{atan}(K_s l(t)^2) + \pi/2 & if\ l \in U \\ -\operatorname{atan}(K_s l(t)^2) + \pi/2 + l(t)^2 & if\ l \notin U \end{cases} \tag{13}$$

where U is the set of all points contained within the support polygon and K_s is a gain coefficient. The arctangent strongly penalizes ZMP positions near the edge of the support polygon, while providing a near constant cost within the interior of the support polygon. K_s serves to shape the gradient of the cost as it approaches the polygon edge (Figure 3). Outside the polygon, the added $l(t)^2$ term dominates and provides a gradient leading back inside the polygon.

We design the stability cost based on the distance between the ZMP and the edge of the polygon, rather than the center of the polygon. The nature of the serial-link robot design lends itself to form polygons that are more ellipsoid / elongated than circular, and our cost function allows the ZMP to move throughout the polygon with relatively equal weighting until approaching the edge, rather than biasing the ZMP to stay within a range of a center point.

The ZMP is a function of the joint angles and so its position in the polygon changes over the course of a trajectory. We therefore optimize the sum of $C_s(t)$ over the entire trajectory.

$$C_s = \begin{cases} \int -\operatorname{atan}(K_s l(t)^2) + \pi/2\ dt & if\ l \in U \\ \int -\operatorname{atan}(K_s l(t)^2) + \pi/2 + l(t)^2\ dt & if\ l \notin U \end{cases} \tag{14}$$

C_p is a *potential* cost. For carrying out the directive goal and remaining stable, C_p can be neglected. However, with knowledge of the environment, we can generate a potential field as in [15] to bias the robot's motion towards (or away from) a desired portion of the workspace. In our results, we show the effectiveness of using a potential field to maneuver the robot into the confined space of a vertical pipe.

The localization problem of relating the snake to the environment is described in [16]. Therefore, we begin the planning problem using the known location and orientation of an obstacle, in this case a pipe, relative to the base of the snake robot. The base reference frame is preserved over the course of motion planning, because the end-most modules in the base do not shift relative to the world.

The potential cost serves to incorporate knowledge of the environment into the cost function. Forming C_p as a

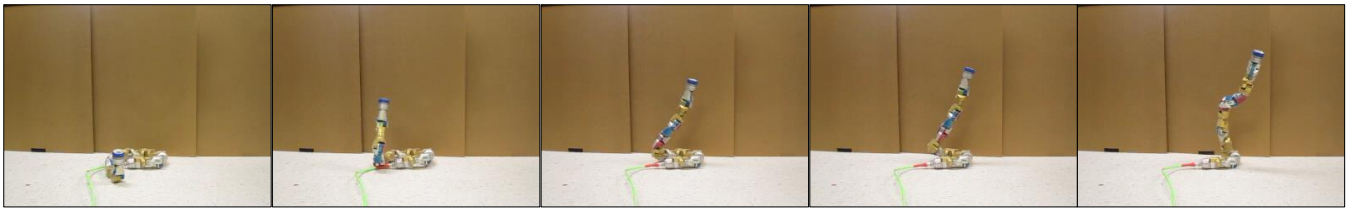

Figure 4. Execution of trajectory plan using the *directive* and *stability* sub-costs.

repelling force around the pipe allows the planner to find solutions that avoid colliding with the obstacle. Here, we have chosen to use C_p as a positive attraction field towards the center of the pipe, to illustrate how the cost can funnel the robot's motion into constrained spaces.

$$C_p = \int \left(\sum_{i=1}^{n} (x_w^i(t) - q_x)^2 - \left(y_w^i(t) - q_y\right)^2 \right) dt \qquad (15)$$

In the above equation, n is the number of active modules and $q_{x,y}$ is the x- and y- position of the center of the pipe. Minimizing this cost therefore reduces the error between the active portion and the center axis of the pipe. As modules are added to the active portion, this naturally funnels the head of the robot towards the pipe and keeps modules that have reached the pipe remaining close to the vertical pipe axis.

The sub costs described above do not explicitly minimize joint torques because the torques resulting from the given cost function were within the limits of the robot used to validate the algorithm. If joint torques are of explicit interest, a torque-specific cost term may be added to the cost function.

E. *Dynamic Weighting, Error checks, and Random Restarts*

The initial planning stage looks only at moving the designated active portion relative to the current support polygon. In order to transition modules from the base to the active manipulator portion, the relative value of the sub-costs changes—it becomes more important to maintain stability while preparing to transition modules than it is to reach the end goal of having the head raised. We therefore implement a dynamic weighting strategy when planning to shift modules from the base to the active portion.

After the initial plan has been made for the designated active modules, if the directive goal is not yet reached, the planner works to transition modules to the active portion to extend the reach. Before transitioning modules, the stability of the robot's current configuration is checked against the next projected stability polygon—the support polygon formed by a base made of two fewer modules. At this point, if the robot needs to alter its position so the ZMP is within the projected polygon, we decrease gain K_1 to reflect the fact that reaching the directive is of less importance than maintaining stability with the future robot configuration. Using the same cost function with the updated gains, the current active-base module designations, and a support polygon defined as the intersection of U and U' as in equation (1), we re-plan using the ending configuration of the last plan as the starting pose for the new plan. Due to the lowered value of K_1, the planner shifts the active portion so the ZMP is closer to the desired polygon. The ending configuration of the plan is once again checked for future stability, and K_1 is decreased and the trajectory re-planned until the stability criteria has been met. K_1 is then reset to its initial value, the module designation is changed to reflect the transfer of modules from the base to the active portion, and the planner iterates.

Gains K_1, K_2, and K_3, shown in equation (9), balance the sub-costs and are application dependent. We found for our system that an equal weighting of K_1 and K_2 produced desirable, stable trajectories for transitioning over half the robot's modules to the active portion without the need for reweighting. Once over half the modules had been transitioned to the active portion, performing a gradient descent on K_1 using a constant step size was sufficient for re-planning with only one to two iterations.

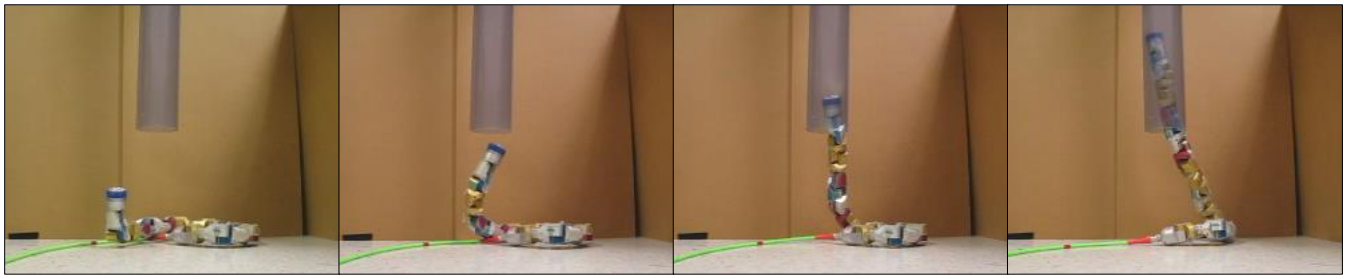

Figure 6. By adding a cost-decreasing potential field, we can direct the motion of the robot as desired. Here, we show the ability of this approach to plan trajectories for the robot head to enter confined spaces by applying a potential field around the known position of a pipe.

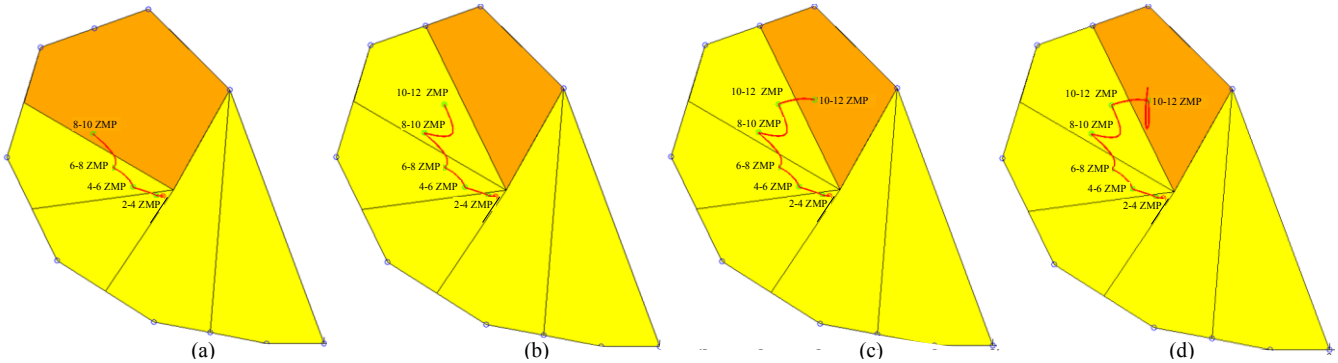

Figure 7. Illustration of the support polygon and the path of ZMP inside the polygon during execution of a trajectory; the position of the ZMP throughout the trajectory is shown in red, the next projected support polygon is shown in orange. In (a), the ZMP when the system has an 8-module active portion is shown inside the next projected support polygon. The system is then stable to move onto a 10-module active portion. In (b), the planner has computed a trajectory that is stable for the current 10-module active potion, but will not be stable in the next support polygon (the ZMP is not inside the orange polygon). (c) shows the planned trajectory to shift the ZMP into the next support polygon. (d) shows the final path of ZMP for the 12-module active portion.

The kinematics of the serial link manipulator result in a non-convex optimization function. Therefore, the optimization step may return an undesirable local minima, depending on the initial values of the spline constants. We also do not employ a hard constraint to force the optimization to return trajectories which keep the ZMP inside the polygon, relying on the strong gradient provided by the cost function to keep the ZMP from exiting the polygon.

We therefore use a combination of a global error check and random restarts to re-plan, should a chosen initial condition result in an undesirable solution. The most important check is the stability check: does the ZMP stay inside the polygon for the total length of the given trajectory? If not, we randomly reseed the spline coefficients over which we optimize and re-plan. If we have added modules to the active portion since the last plan, we do a further check to confirm that the height of the head module is above the previous head height. If it is not, we reseed using random values and re-plan.

IV. Experimental Results

We illustrate the effectiveness of this algorithm by showing capabilities enabled using the described planning framework.

Figure 4 shows the snake balancing three-quarters of its body above the remaining modules in order to raise its head, containing the camera, as high as possible for surveillance and inspection tasks. For this experiment, a potential cost was not considered and only the directive and stability costs were used. This illustrates the ability of this algorithm to effectively lift modules from the support phase to the active phase and maintain stability.

We illustrate the effectiveness of directing the robot with the addition of C_p based on the location of a vertical pipe as described in Section D, and as shown in Figure 6. To execute this maneuver, the planner iteratively plans to transition 2-10 modules from the support phase to the active phase. After planning for ten modules, the planner finds that the ten-module active manipulator, with seven support modules, will be unstable should the planner continue on to a twelve module active manipulator with five support modules. The planner successfully shifts the ten-module active portion so that the current ZMP of the system is within the current support polygon and the next projected support polygon. The planner then successfully transitions the modules and finishes planning for a 12-module active manipulator portion to stably reach as high as possible. The position of the ZMP in relation to the current and projected support polygons throughout the trajectory is shown in Figure 7. The addition of the potential field based on the pipe directs the head towards the mouth of the pipe as the robot raises modules, and minimizes error between all active modules and the pipe axis, allowing the robot to maneuver into the pipe.

V. Conclusion and Future Work

The FBSR planning method shown here is robust and returns stable solutions relevant to the environment which greatly extends the capabilities of floating base serial link robots. Using random starts and recursively optimizing over a weighted cost function, we can dynamically change the weights of the relative costs to better reflect the priorities of the current task to robustly return smooth, stable joint trajectories for executing balance critical tasks. For tasks such as those demonstrated here, optimizing the constants of cubic spline trajectories, per joint, is a tractable and valid method of generating trajectories. For tasks necessitating additional constraints on the splined trajectories, splines of higher order can be used or knot points added to increase the available degrees of freedom over which to search. However, the planning time increases with the addition of optimization parameters—either through the introduction of knot points, higher order splines, or the addition of joints. Therefore, future work includes examining the feasibility of optimizing parameterized backbone curves rather than cubic splines, to maintain a constant number of optimization parameters.

References

[1] M. Tesch, K. Lipkin, I. Brown, R. Hatton, A. Peck, J. Rembisz, and H. Choset, "Parameterized and Scripted Gaits for Modular Snake Robots," *Advanced Robotics*, vol. 23, no. 9, pp. 1131–1158, Jan. 2009.

[2] B. Y. S. Hirose and H. Yamada, "Snake-Like Robots," March, pp. 88–98, 2009.

[3] A. A. Transeth, K. Y. Pettersen, and P. Liljebäck, "A survey on snake robot modeling and locomotion," *Robotica*, vol. 27, no. 07, pp. 999–1015, 2009.

[4] R. L. Hatton and H. Choset, "Sidewinding on slopes," *2010 IEEE International Conference on Robotics and Automation*, pp. 691–696, May 2010.

[5] M. Tesch, K. Lipkin, I. Brown, R. Hatton, A. Peck, J. Rembisz, and H. Choset, "Parameterized and Scripted Gaits for Modular Snake Robots," *Advanced Robotics*, vol. 23, no. 9, pp. 1131–1158, Jan. 2009.

[6] L. Righetti, J. Buchli, M. Mistry, and S. Schaal, "Inverse dynamics control of floating-base robots with external constraints: A unified view," *2011 IEEE International Conference on Robotics and Automation*, pp. 1085–1090, May 2011.

[7] M. Vukobratović and B. Borovac, "Zero-Moment Point — Thirty Five Years of Its Life," *International Journal of Humanoid Robotics*, vol. 01, no. 01, pp. 157–173, Mar. 2004.

[8] E. Yoshida, O. Kanoun, C. Esteves, and J. P. Laumond, "Task-driven Support Polygon Reshaping for Humanoids," *2006 6th IEEERAS International Conference on Humanoid Robots*, pp. 208–213, 2006.

[9] M. Yamakita, M. Hashimoto, and T. Yamada, "Control of locomotion and head configuration of 3D snake robot (SMA)," *2003 IEEE International Conference on Robotics and Automation*, vol. 2, pp. 2055–2060, 2003.

[10] M. Tanaka and F. Matsuno, "Modeling and Control of Head Raising Snake Robots by Using Kinematic Redundancy," *Journal of Intelligent & Robotic Systems*, pp. 1–5, Jul. 2013.

[11] M. Nilsson, "Snake robot free climbing," *Proceedings of International Conference on Robotics and Automation*, vol. 4, pp. 3415–3420, 1997.

[12] C. Ye, S. Ma, B. Li, and Y. Wang, "Head-raising Motion of Snake-like Robots," *2004 IEEE International Conference on Robotics and Biomimetics*, pp. 595–600, 2004.

[13] C. Wright, A. Buchan, B. Brown, J. Geist, M. Schwerin, D. Rollinson, M. Tesch, and H. Choset, "Design and architecture of the unified modular snake robot," *2012 IEEE International Conference on Robotics and Automation*, pp. 4347–4354, May 2012.

[14] R. M. Murray, Z. Li, and S. S. Sastry, *A mathematical introduction to robotic manipulation*. CRC press, 1994.

[15] H.Choset, ed. *Principles of robot motion: theory, algorithms, and implementations*. MIT press, 2005.

[16] H. Ponte, M. Queenan, C. Gong, C. Mertz, M. Travers, F. Enner, M. Hebert and H. Choset. "Visual Sensing for Developing Autonomous Behavior in Snake Robots," *2014 IEEE International Conference on Robotics and Automation.*