Fast, Sampling-Based Kinodynamic Bipedal Locomotion Planning with Moving Obstacles

Junhyeok Ahn, Orion Campbell, Donghyun Kim and Luis Sentis

I. SUMMARY

In this paper, we present an algorithm for sampling-based kinodynamic planning and control for a bipedal robot in complex environments. We plan dynamically and kinematically consistent footstep placements using a kinodynamic RRT with a steering function customized for a biped. We exploit a novel metric function instead of using the Euclidean metric which is not related to the robot's dynamics. In the procedure, we take advantage of the analytic solutions of the Linear Inverted Pendulum Model (LIPM) in the phase space to obtain the proper footstep placement and the full state of the Center of Mass (CoM) for each step. We demonstrate this algorithm in a complex simulation environment which includes both static and moving obstacles with a human-sized humanoid robot, Valkyrie.

II. INTRODUCTION

A high priority in the field of bipedal robotics is to develop algorithms that reliably produce robust and agile dynamic locomotion that achieves a given mission in a complex environment. However, the computational complexity of high-dimensional, dynamically-constrained motion planning for humanoids has limited the success of many planning algorithms that are popular for other applications.

This paper presents a novel bipedal locomotion planning and control framework that simultaneously addresses kinematic feasibility, obstacle avoidance, dynamic consistency, and an array of full state start and goal specifications for a high-dimensional humanoid robot via a kinodynamic RRT algorithm. We propose a steering method and a pseudo-metric for the RRT planner that exploits the analytic solutions in phase space for switching time and footstep placements, and a controller design that stabilizes centroidal momentum to allow a biped to reliably maneuver in a complex, dynamic environment.

III. PHASE-SPACE LOCOMOTION CONTROL

We present analytic solutions in phase space to calculate lateral directional footstep and swithcing time from given sagittal directional footstep and velocity, and exploit these in our kinodynamic planning in section IV.

Our overall algorithm can be summarized by Algorithm 1. P_1, $\mathbf{X}_1$, $\mathbf{Y}_1$, $\mathbf{X}_{apex,2}$ and $\mathbf{X}_{switch}$ refer to $(x_{p,1}, y_{p,1})$, $(x_1, \dot{x}_1)$, $(y_1, \dot{y}_1)$, $(x_{apex,2}, \dot{x}_{apex,2})$ and $(x_{switch}, \dot{x}_{switch})$. Subscript switch means intersection point of discontinuous dynamic in phase space, t_{switch} and t_{apex} mean time duration from apex to intersection point and from intersection to next apex point.

Algorithm 1 Computation of t_{switch}, y_p

Require: $P_1,\ \mathbf{X}_1,\ \mathbf{Y}_1, \dot{y}_{apex,2}\ \mathbf{X}_{apex,2}$

1: $\mathbf{X}_{switch} \leftarrow$ FIND_X_SWITCHING_STATE$(x_{p,1}, \mathbf{X}_1, \mathbf{X}_{apex,2})$
2: $t_{switch} \leftarrow$ GET_TIME_AT_STATE$(x_{p,1}, \mathbf{X}_{swtich})$
3: $t_{apex} \leftarrow$ GET_TIME_AT_STATE$(x_{p,1}, \mathbf{X}_{apex,2})$
4: $\mathbf{Y}_{switch} \leftarrow$ INTEGRATION$(y_{p,1}, \mathbf{Y}_1, t_{switch})$
5: $y_p \leftarrow$ FIND_YP$(\dot{y}_{apex,2},\ \mathbf{Y}_{switch}, t_{apex})$
6: **return** (t_{switch}, y_p)

IV. KINODYNAMIC RRT PLANNING

We define a vertex V_i in the RRT tree to be a tuple with a configuration q_{apex} and an LIPM apex state $\mathbb{S}_{\text{LIPM}}$ which includes the time t_{clock} that the CoM will pass through that vertex. We define the configuration space to represent the global position and longitudinal direction of motion of the biped.

First, we randomly sample q_{apex} in our configuration space with given parameters. Then, we specify our $q_{apex}s$ between tree nodes and q_{sample} along Dubin's path. We calculate corresponding $\mathbb{S}_{\text{LIPM}}$ from algorithm in Section III. According to newly defined metric function which is sum of t_{switch} and t_{apex}, we choose the nearest neighbor. After that, we check collision and we prune the $q_{apex}s$ after collision happen. Our algorithm can be summarized by Algorithm 2.

RRT vertex: $V := (\ q_{apex}\ ,\ \mathbb{S}_{\text{LIPM}}\ )$

config. space: $q_{apex} := (\ x_{glob}\ ,\ y_{glob}\ ,\ \theta_{glob}\ )$

LIPM state: $\mathbb{S}_{\text{LIPM}} := (\ \mathbf{X}_{apex}\ ,\ \mathbf{Y}_{apex}\ ,\ P\ ,\ t_{\text{clock}}\ )$

pivot location: $P := (\ x_{pivot}\ ,\ y_{pivot}\ )$

time duration: $t_{clock} := (\sum_{n=0}^{n} t_{switch}\ +\ t_{apex})$

boundary: $\mathcal{B} := (\ q_{min},\ q_{max}\ )$

$q_{min}.\theta\ = 0,\ q_{max}.\theta\ = 2\pi$

initial vertex: $V_0 := (\ q_{init}\ ,\ \mathbb{S}_{\text{LIPM},init}\ )$

goal vertex: $V_{goal} := (\ q_{goal}\ ,\ \mathbb{S}_{\text{LIPM},goal}\ )$

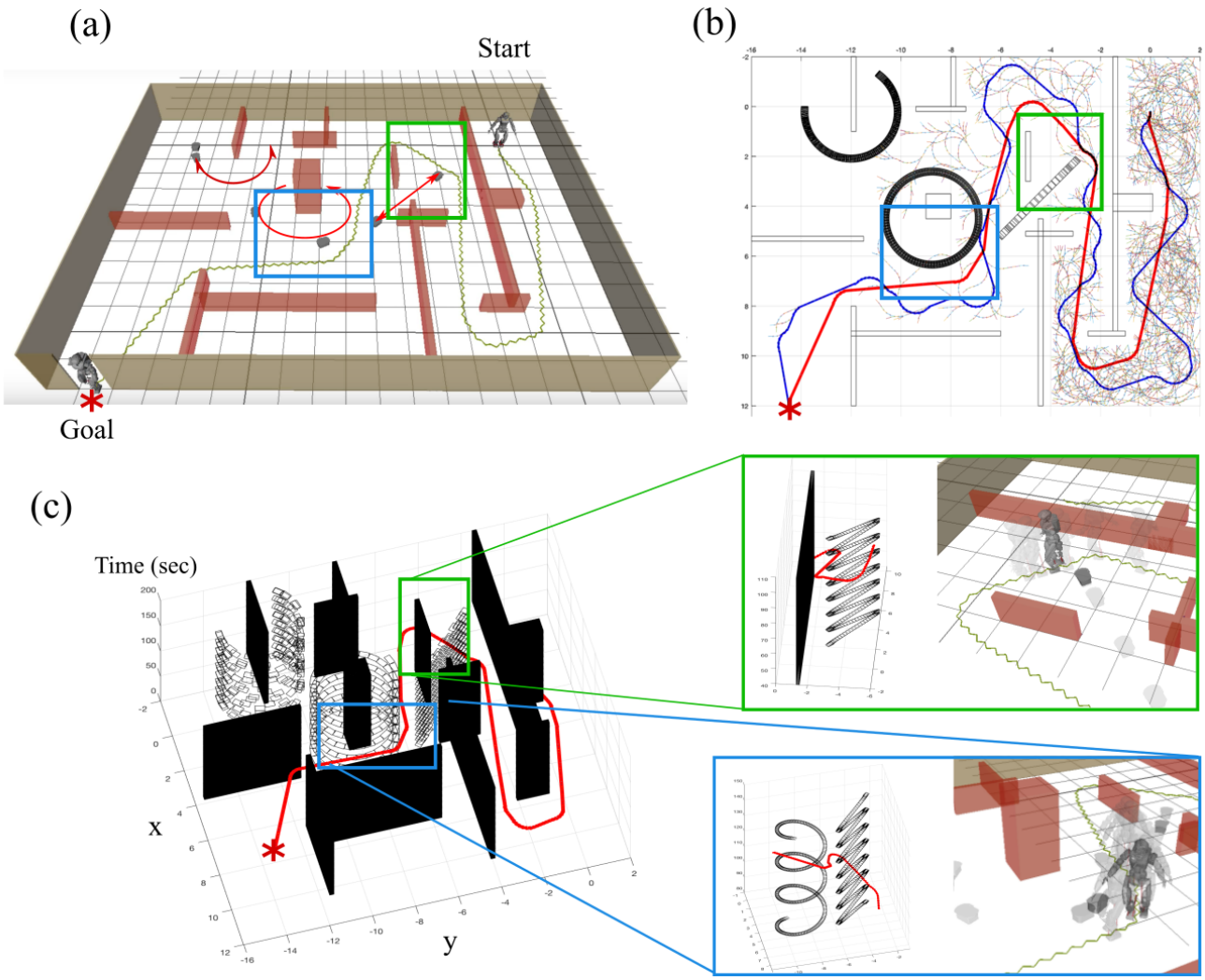

Fig. 1. **Environment Setup and Planned Path.** (a) the room is separated by red sections and three mobile robots are moving around with a regular speed. (b) After the RRT planner finds the solution path (blue), the short cutting process smooths out the path (red). (c) When we plot the path in three dimensional space with time axis, we can clearly see that the CoM path has enough space from the moving obstacles. Over three figures, the locations indicated by green and blue squares are identical

V. Result

In the simulation, Valkyrie starts to walk on the right upper corner of a 18×14 m-size room. (Fig. 1(a)). The proposed planner successfully finds the solution route to the goal location in 120 s at most (Fig. 1(b)).

The time along the planned path is important to check the collision with moving obstacles. Our kinodynamic planner provides collision free path for not only static but also dynamic obstacles. Fig. 1(c) shows that the CoM path and moving obstacles are far enough in any time along the path, which secures that the stance foot does not interfere with the obstacles.

Algorithm 2 Kinodynamic RRT Footstep Planner

Require: $\dot{x}_{apex},\ \Delta x_{p,max},\ \rho_{min},\ \mathcal{B},\ k,\ obs\_list$
procedure BUILD_RRT(n, V_0, q_{goal})
  Tree.INIT(V_0)
  $param \leftarrow (\dot{x}_{apex},\ \Delta x_{p,max},\ \rho_{min},\ k)$
  for $i \leftarrow 1, n$ **do**
    $q_{\text{RS}} \leftarrow$ RANDOM_SAMPLE($\mathcal{B}, q_{goal}$)
    $V_{\text{NN}} \leftarrow$ NEAREST_NEIGHBOR(*Tree*, q_{RS}, *param*)
    Path $\leftarrow$ CONNECT($V_{\text{NN}}, q_{\text{RS}}$, *param*)
    Path $\leftarrow$ PRUNE_FOR_COLLISION(*Path*, $\mathcal{B}$, *obs_list*)
    Tree.APPEND_PATH_TO_TREE(V_{NN}, *Path*)
    if (*Path.end*().q == q_{goal}) **then**
      return *Tree*.EXTRACT_PATH(V_0, *Path.end*())
    end if
  end for
  return *Failure*
end procedure