Push Recovery and Online Gait Generation for 3D Bipeds with the Foot Placement Estimator

Brandon J DeHart and Dana Kulić

***Abstract*— Humanoid robots have many potential applications in man-made environments, including performing hazardous tasks, assisting the elderly, and as a replacement for our aging workforce. However, generating a reliable gait for biped robots is challenging, particularly for dynamic gait and in the presence of unknown external disturbances, such as a bump from someone walking by. In this work, a 3D formulation of the Foot Placement Estimator is used with a high-level control strategy to achieve a dynamic gait capable of handling external disturbances. A key benefit of this approach is that the robot is able to respond in real time to external disturbances regardless of whether it is at rest or in motion. This strategy is implemented in simulation to control a 14-DOF lower-body humanoid robot being subjected to unknown external forces, both when at rest and while walking, and shown to generate stabilizing stepping actions.**

I. Introduction

In the realm of humanoid robotics, one of the major challenges is the ability for bipedal robots to maintain balance during gait. This is especially difficult when the biped is subject to some form of external disturbance, which may move it away from any precomputed trajectories or planned motions. This problem is of critical importance if bipeds are to be useful as part of our regular daily lives, where they may experience random external disturbances frequently.

Currently, a number of different approaches exist when generating and carrying out biped gait, with the vast majority using the Zero Moment Point (ZMP) [1] as a control reference to generate trajectories either offline [2] or online [3]. The desired ZMP trajectory can be modified in response to external disturbances [4]. The ZMP is the Center of Pressure (COP) of a biped: the point within the convex support polygon at which the vertical ground reaction force acts [5]. Although it is used in many humanoid control applications, the ZMP is only useful and valid when a robot is statically stable, leading to a reliance on a static gait, where the biped is able to stop at any time during the gait cycle without falling over. This requires the introduction of measures of stability, such as the Foot Rotation Indicator [5], to measure the level of stability and attempt to maintain it continuously. In this work, the ZMP (referred to hereafter as the COP) is used only during static portions of dynamic gait generation.

An alternative formulation to the gait generation problem is to focus on where the robot needs to step next to restore balance. An example of this approach is the instantaneous Capture Point [6], and a later extension to the Capturability regions [7], [8]. For a hopping or running robot, energy conservation can be used to find a desired foot placement [9]. Another approach in this vein is the Foot Placement Estimator (FPE), originally proposed in [10], and extended in [11]–[14]. The FPE is a point on the ground where a biped would place a swinging foot in order to come to rest above that point, having converted all of its kinetic energy into potential energy. The FPE is computed using the conservation of angular momentum, finding the location where the total energy of the biped after swing foot impact is equal to the peak potential energy. The 'Capture' methods use the 3D Linear Inverted Pendulum Model [15], which assumes that the Center of Mass (COM) of the biped remains at a fixed height and the impact of foot placement does not affect the COM position and velocity. The errors introduced by these assumptions must be actively controlled, requiring a more complex control strategy than the approach in this work based on the FPE and its extensions.

Both the 2D FPE [10] and the Foot Placement Indicator (FPI) [11] are only defined for planar bipeds, with the FPI including the dynamics of articulated legs to extend the original single point mass model of the 2D FPE. The other three extensions of the 2D FPE are all developed in 3D, each with a different approach. A fixed plane is chosen, requiring separate out-of-plane control of balance, in [12]. In [13], the "3D FPE" is defined, based on an "Euler pendulum" monoped model with a disc foot, such that their model is a direct 3D extension of the planar compass biped in [10]. Finally, [14] defines the Generalized FPE (GFPE) using a rimless spoked wheel model, which includes interesting predictive aspects and can handle non-level ground, but is only used to recover from external disturbances while at rest with no discussion of its use while walking.

In this work, the methods in [13] to calculate the 3D FPE are used to determine the FPE point for a 14-DOF lower-body humanoid robot. This point is then used as a control reference to inform both a high-level state machine and its associated task-level trajectory generator. The task-level trajectories are used as control inputs to a prioritized Jacobian-based feedback loop [12] and a simple low-level PD joint controller to drive a simulated robot.

The goal of this overall control strategy is to allow the robot to respond directly to external disturbances by stepping onto the FPE, either while standing still or walking. One of the key benefits of this strategy is that recovery from external disturbances is simply a subset of the elements required for dynamic gait. To walk, the robot needs only to push itself

*This work was supported by the Natural Sciences and Engineering Research Council and the Ontario Graduate Scholarship Program.

Brandon J DeHart and Dana Kulić are with the Department of Electrical & Computer Engineering, University of Waterloo, Ontario, Canada {bjdehart, dana.kulic}@uwaterloo.ca

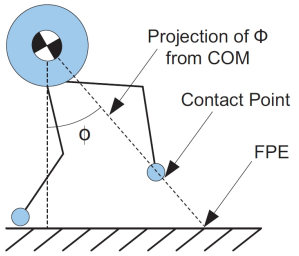


Fig. 1. Illustration of the 2D FPE [12].

into an unstable state in the desired direction of motion, then allow the recovery mechanism to prevent it from falling.

Both [10] and [12] have investigated earlier versions of this form of state machine based control strategy. However, the first deals only with planar bipeds with point feet, while the second uses a constant plane of motion, requires separate lateral stabilization, and has very limited discussion of external disturbances. This work uses a continuously updating plane, similar to [13] and [14], which allows it to respond in 3D to unknown external disturbances. It also includes dynamic gait generation, unlike [14], and is used directly as part of a unified control method, as opposed to a measurement method such as in [13].

This paper's main contribution is the use of the 3D FPE in combination with a continuously updated FPE plane to achieve disturbance compensation both while standing and while in motion. The 3D FPE has only been used until now in the biomechanics literature, and no existing work has been done using a continuously updating FPE plane. This paper demonstrates that this approach is capable of compensating for external, unknown disturbances as part of a gait cycle.

II. Proposed Approach

In this work, the proposed method of handling recovery from external disturbances, and by extension dynamic walking, builds on the previous FPE research in [10], [12]–[14]. These different approaches are explained in further detail below to help justify the choice of the 3D FPE approach to calculate the FPE in 3D, to be used as a control input to a high level state-machine based controller with online task-level trajectory generation and a prioritized Jacobian-based feedback loop, as described in the next section.

The original 2D FPE concept was developed for planar bipeds in [10], with a point mass at the system's Center of Mass (COM), point feet, and massless legs. The 2D FPE is illustrated in Fig.1. The FPE is computed using the conservation of angular momentum in the 2D plane, by finding the angle ϕ where the total energy of the biped after swing foot impact is equal to the peak potential energy. In both [13] and [14], extensions of the 2D FPE concept are made to take 3D dynamics into account. In both approaches, the 3D dynamics are projected into a vertical plane which passes through the COM, but the computation of the plane orientation about the vertical axis differs between the two methods. Once the plane is found, the original 2D FPE algorithm is used, either directly or indirectly depending on the model, to find a point on the line where the plane intersects the ground for the 3D biped to step.

In [14], the GFPE plane direction is defined by the horizontal velocity vector of the COM immediately after an external disturbance. In this formulation, the robot is assumed to be at rest before any external disturbance occurs, and therefore should only move in the direction of the COM velocity vector until action is taken. This assumption, and therefore the plane itself, is reasonably valid if the robot is standing still when a disturbance takes place, but is less appropriate if the robot is already moving when disturbed (such as being bumped while walking). In [13], the 3D FPE plane direction is defined by a horizontal vector perpendicular to the overall angular momentum of the robot, taken about the ground projection of the COM.

The system model and the robot's state are used to calculate the FPE planes. In the equations below, the following conventions are used: $i = 1..N$ where N is the number of links, m_i is the mass of each link, M is the total mass of the system ($\sum m_i$), c_i and v_i are the absolute position and velocity of the COM of each link (relative to the origin, O), ω_i is the absolute angular velocity of link i, R_i is the rotation matrix of link i relative to the origin, and I_i is the rotational inertia matrix for link i around c_i in local coordinates.

The COM (c) is given by

$$c = \frac{\sum m_i c_i}{\sum m_i} = \frac{\sum m_i c_i}{M} \tag{1}$$

while the ground projection of the COM (c_{GP}) is

$$c_{GP} = c - (c \cdot \hat{k})\hat{k} \tag{2}$$

The linear momentum (P) of the system is

$$P = \sum P_i = \sum m_i v_i \tag{3}$$

while the velocity of the COM (v_c) is

$$v_c = \frac{dc}{dt} = \frac{\sum m_i v_i}{\sum m_i} = \frac{P}{M} \tag{4}$$

The angular momentum of the system is calculated about two different points: the origin (H_O) and the ground projection of the COM (H_{GP}), given in (5) and (6), respectively.

$$H_O = \sum (c_i \times P_i + R_i I_i R_i^T \omega_i) \tag{5}$$

$$H_{GP} = H_O + c_{GP} \times P \tag{6}$$

As shown in equation (5), the angular momentum of the system takes into account both the linear and angular velocities, and the rotational inertia of all of the links in the robot. It is also clear from equation (6) that the angular momentum calculated about the COM ground projection directly takes into account the velocity of the COM, in addition to the elements included in equation (5).

This means that the GFPE plane defined in [14] can be considered a special case of the 3D FPE plane, where none of the links are rotating or moving relative to one another. Since this special case is only valid when the pre-disturbance robot is at rest or translating purely linearly (difficult to achieve in

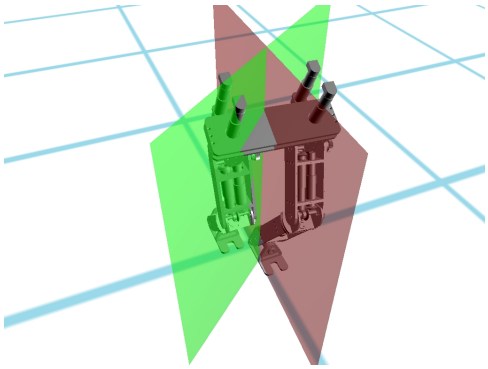


Fig. 2. Image showing the planes defined in [14] and [13]. The robot's right foot (on the left side of the image) has just lifted during walking. The green plane is the 3D FPE plane, while the red plane is the GFPE plane.

the real world), the plane from the 3D FPE work is used in the remainder of this work. This allows the robot to respond to disturbances not only in the restricted cases of the GFPE plane, but also when it is already moving.

The difference between the two planes is shown in Figure 2, just after the robot has lifted its right foot (on the left, in the figure) while walking: the 3D FPE plane has already responded to this change by rotating towards the lifted foot, but the GFPE plane remains aligned with the current direction of linear translation. Once the robot starts to fall towards the lifted foot, the GFPE plane approaches the 3D FPE plane, as the COM begins to move in that direction.

Once the horizontal direction of the plane has been calculated, a number of other values must be calculated and then projected onto the plane, to determine the position of the FPE within the plane. These equations all make use of a unit vector normal to the chosen plane, labeled $\hat{n}$, which in the case of the 3D FPE plane is simply the horizontal direction of H_{GP}.

First, the angular velocity (ω_i) and rotational inertia (I_i) of each link needs to be projected into the plane:

$$\omega_i^{\hat{n}} = (R_i \omega_i) \cdot \hat{n} \tag{7}$$

$$I_i^{\hat{n}} = \hat{n}^T \cdot (R_i I_i R_i^T) \cdot \hat{n} \tag{8}$$

Second, the distance (d_i) between each link's COM (c_i) and the system COM (c) is found in the plane:

$$d_i = c_i - c - ((c_i - c) \cdot \hat{n})\hat{n} \tag{9}$$

Third, the system's average angular velocity ($\omega^{\hat{n}}$) and total rotational inertia ($I^{\hat{n}}$) are needed:

$$\omega^{\hat{n}} = \sum I_i^{\hat{n}} \omega_i^{\hat{n}} / \sum I_i^{\hat{n}} \tag{10}$$

$$I^{\hat{n}} = \sum I_i^{\hat{n}} + m_i d_i \tag{11}$$

Finally, the FPE point is found using a modified form of the 2D FPE equations from [13]. First the angle ϕ between the vertical axis and a line between the COM and the FPE is found via equation (12). In this equation, h is the height of the COM ($h = c^{\hat{k}} = c \cdot \hat{k}$), $\dot{h}$ is the rate of change of the height ($\dot{h} = dh/dt$), and $v_c^{\hat{u}}$ is the velocity of the COM in the chosen plane ($v_c^{\hat{u}} = v_c \cdot \hat{u}$, where $\hat{u} = \hat{k} \times \hat{n}$).

$$\frac{[Mh(\dot{h}\sin\phi + v_c^{\hat{u}}\cos\phi)\cos\phi + I^{\hat{n}}\omega^{\hat{n}}\cos^2\phi]^2}{2Mgh\cos\phi(1-\cos\phi)(Mh^2 + I^{\hat{n}}\cos^2\phi)} = 1 \tag{12}$$

Once the angle ϕ is found, the FPE is found as follows:

$$FPE = c_{GP} + h\tan\phi\hat{u} \tag{13}$$

This position on the ground is used as a reference position for the high-level state machine and the task-level trajectory generation algorithms described in the next section. More details on the development of the original 2D FPE and the 3D FPE extension can be found in [10] and [13].

III. Controller

A gait controller is defined in this section which enables a biped to recover from external disturbances, both at rest and while walking, using the 2D gait controller from [10] as a starting point. Although a 3D extension of the original controller was made in [12], it used a constant FPE plane in the desired direction of motion, requiring the use of ZMP-based stable gait generation for the majority of the gait cycle to compensate for the out-of-plane 3D dynamics. The existing controller was augmented to improve the walking performance, both by defining functions at a higher level and including states for leaning forward (to instigate walking) and settling the feet on the ground.

The high-level controller is built around the use of a symmetric state-machine which defines the high level state of the robot at any particular time, as shown in Fig. 3. A key feature of the controller is that recovery from external disturbances while at rest is a direct subset of the functionality required to walk. To walk, the high-level controller simply needs to cause the robot to move its COM until it begins to fall in the desired direction of motion, then enter push-recovery mode. It should be noted that the COM moving outside the robot's support polygon is only indirectly related to this mode: If the angular momentum is large enough, the stepping strategy will kick in before the COM exits the support polygon.

The state of the system, as determined by this state machine, is then used to generate task-level trajectories, for the COM and the position and orientation of both feet, to achieve the objectives of the given state. These trajectories are generated online, to avoid the need to calculate trajectories a priori and therefore allow maximum flexibility to respond to disturbances. Depending on the current state, different objectives exist for the desired COM and position and orientation of each foot. A set of goal positions and orientations are defined, based in part on the positions and orientations when each state starts. A trajectory is then generated using a quintic spline, to allow specification of the trajectory's initial and final derivatives.

To translate between the task-level trajectories and the low-level desired joint angles of the robot, a prioritized Jacobian-based feedback loop is used. This portion of the control method has been described previously in [12], so will only be briefly introduced here. In each state, the task

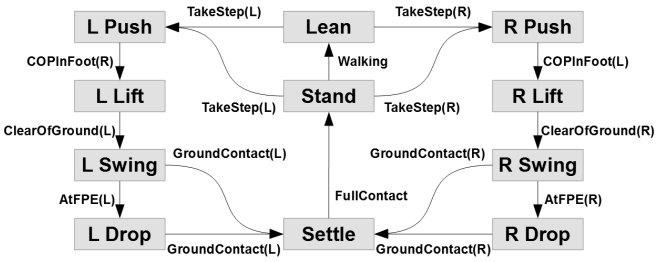


Fig. 3. Diagram of the high-level state machine used in the controller.

trajectories are prioritized into high and low categories. The high- and low-priority Jacobians are then built by stacking the individual task Jacobians in each priority category. The overall Jacobian is found by projecting the low-priority Jacobian onto the null space of the high-priority Jacobian, and this overall Jacobian is inverted to translate between the task trajectories and desired joint trajectories. Finally, a simple PD control loop is used to calculate the torques required to move the robot to the desired joint angles.

A. State Machine

In the original state machine in [10], developed for a simple planar robot, the states defined the joint movement trajectories directly. These definitions were specific to the particular biped geometry, and not easily generalizable to different robot configurations. In [12], the trajectories were defined at the task space level, in terms of Cartesian trajectories of the COM and swing foot, providing a more general framework for arbitrary DoF legs. However, the trajectories defined in [12] were still specific to the a-priori chosen FPE plane and assumed perfectly level ground.

In the state machine described below, the states and the transitions have been defined with a further level of abstraction, to facilitate its use on a broad range of different biped robots and handle on-line changes to the orientation of the FPE plane. For example, during either of the 'Swing' states, the swinging foot is required to maintain "ground clearance" instead of "a height of at least h", to allow for variable terrain in the future.

1) States and Objectives: In these descriptions, the F represents either the left (L) or right (R) foot:

- *Stand*: The COM is moved to a set height to maximize its ability to respond to disturbances, centered between the positions of the two feet. The second objective is to maintain full ground contact with both feet.
- *Lean*: Move the COM at the desired COM velocity (where a velocity of 0 is equivalent to standing still), while maintaining full ground contact with both feet.
- *F Push*: Move the COP into the opposite foot, by augmenting the desired COM velocity.
- *F Lift*: Lift and rotate the given foot off the ground to achieve the desired ground clearance, while using remaining joints to maintain the desired COM velocity.
- *F Swing*: Move the given foot horizontally towards a point above the FPE, while both maintaining ground clearance of the foot and the desired COM velocity.
- *F Drop*: Lower the given foot onto a tracking point based on the FPE, while maintaining the desired COM velocity. The tracking point is offset from the FPE depending on which mode the simulation is in. If the robot is walking, the tracking point is slightly behind the FPE, allowing the robot to maintain momentum in the desired gait direction. If the robot is standing and responding to a disturbance, or intends to stop walking, the tracking point is slightly in front of the FPE.
- *Settle*: Maximize the ground contact area of both feet.

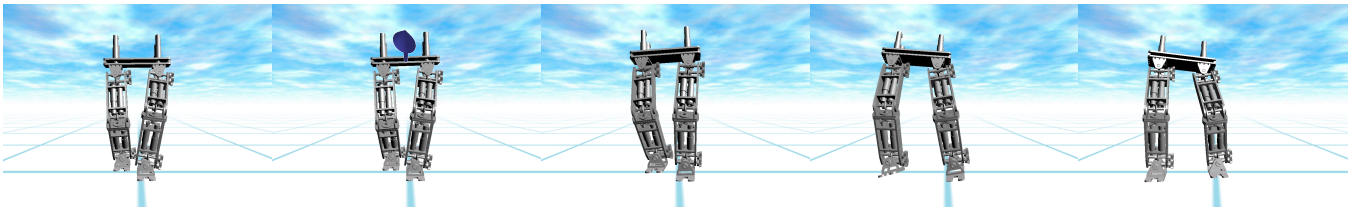
2) Transition Functions:

- *TakeStep(F)*: Checks if the given foot should take a step. The function is active if the distance between the FPE and the given foot is larger than that between the feet.
- *COPInFoot(F)*: Checks if the Center of Pressure is within the given foot. The function is active once the COP is at least a set distance within the foot's perimeter.
- *ClearOfGround(F)*: Checks if the given foot is clear of the ground. The function is active if the entire foot contact surface is a suitable distance away from any portion of the ground contact surface.
- *AtFPE(F)*: Checks if the given foot has reached a point above the FPE. The function is active if the given foot position is within a set horizontal distance of the FPE.
- *GroundContact(F)*: Checks if the given foot has touched the ground. The function is active if the foot's contact surface has touched the ground's contact surface.
- *FullContact*: Checks if both feet are in full contact with the ground. The function is active when both feet are flat on the ground.
- *Walking*: Checks if there is a non-zero desired COM velocity. If so, the function is active.

IV. Simulations and Results

The proposed approach was verified in simulation using a model of a 14 DoF lower body humanoid robot (700mm tall, 30kg) developed at the University of Waterloo. The robot has 7 DoF in each leg, 3 at the hip, one at the knee, and 3 at the ankle. The CAD drawings of the robot were used to generate a SimMechanics dynamic model of the robot, including models for the robot and motor dynamics [16]. Various simulations were carried out where internal (walking) and external (pushes) disturbances were introduced, and the biped used the proposed controller to regain stability following a disturbance. These simulations consist of three classes of disturbance: being pushed while standing still, dynamic walking, and being pushed while walking. In all of these simulations, the robot was initialized to begin the simulation as if it had just taken a step while walking (i.e. with one foot ahead and to the side of the other).

The first simulation was used to verify that the controller was acting as expected in the presence of an external disturbance while the robot was stationary. A simulated force of 150 N was applied to the center of the torso in a direction out of the page for 1/10 second. The series of images in Figure 4 show the progression of the simulation at the transitions between the various states in the controller's state machine.



(a) Initial Pose (b) After Being Pushed (c) After Lift state (d) After Swing state (e) In Stand state

Fig. 4. Simulation results when the robot is pushed from behind on the center of the torso with a force of 150 N for 0.1 seconds. The dark blue circle indicates the location of the disturbance force, the orientation is perpendicular to the plane shown, out of the page.

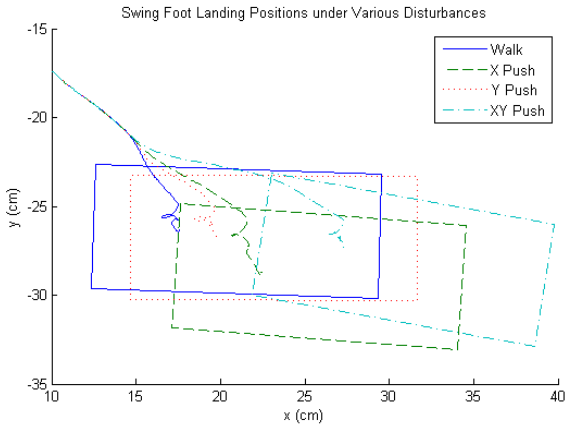


Fig. 5. Landing positions of the swinging foot in each of the four walking scenarios: normal walking (Walk), and pushes (* Push) in the X, Y, and XY directions. The foot paths after being pushed are also shown from the point they start to differ, using the same notation.

The second simulation included a desired COM velocity, to dictate how fast and in what direction the robot should attempt to walk. In this simulation, the robot is statically stable until it reaches the Swing state, as the Lift state is augmented to move the swinging foot in the direction of desired motion before switching to tracking the FPE in the Swing state. A series of images, in Figure 6, shows the simulation at the transition points between states for one step of the walking cycle.

In the final set of simulations, the robot was subjected to external disturbances from various angles while dynamically walking. For these simulations, the disturbance lasted for 0.1 seconds and acted on the center of the torso, as in the standing simulation. The external force was applied during the Swing state, to demonstrate the robot is capable of responding to disturbances while in the dynamic portion of its walk cycle. The landing positions of the swinging foot in each of the various scenarios are shown in Figure 5, along with the foot paths from the point they start to differ.

In Figures 7a and 7b, the trajectories of the X and Y positions of the FPE in the three different push-while-walking scenarios are compared to the trajectory of the FPE in the normal walking scenario. Since the FPE is located on the ground, and the ground is assumed to be flat, the Z component is left out of the comparisons.

It is apparent from these graphs that at the time of the push (2 s), the FPE position starts to differ dependent on the different forces applied. The main thing to note in these graphs is that a push in the X direction creates much more of an impact in the FPE's Y position than a push in Y, and vice versa. Another key observation is that a push in the X direction appears to produce a delay in the movement of the FPE's X position, similarly in Y as well.

As can be seen from the final robot poses in Figures 4, 6, and the attached video, the controller performs very well in all applied scenarios. All of the final poses are statically stable, and in the case of the walking simulations the robot moves on to begin a step with the opposing foot.

V. Discussion

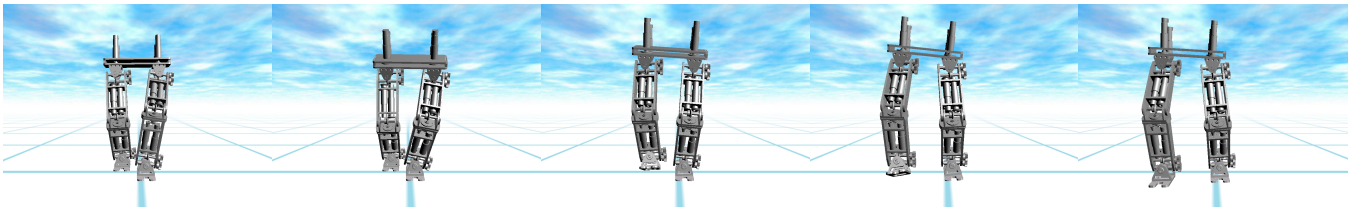
Some potential issues with the current state machine are:

- If the robot is pushed in the F Lift or F Swing states, the wrong foot may be in the air. However, the continuous calculation of the FPE should allow the controller to adapt to many of these disturbances. For example, if the robot is pushed from the side opposite to the lifted foot, the raised foot will attempt to track the new FPE. If it is unable to do so, then it will make contact with the ground and a new round of push recovery will begin with the other foot.
- If the robot is pushed away from the FPE in the F Drop state, it is possible that the robot will fall. A possible solution is adding a transition to detect this and switch back to the F Swing state.
- If both feet are an equal distance from the FPE, the chosen foot depends on the order of transition evaluation.

In the specific implementation used in the simulations above, the Push and Lean states were combined to allow the robot to both move in the desired direction of motion and shift its COP into the chosen stance in parallel. This may have affected the timing of the initial states while walking, but since these states are during statically stable double support sections of the gait, this would not have affected the overall performance of the simulations.

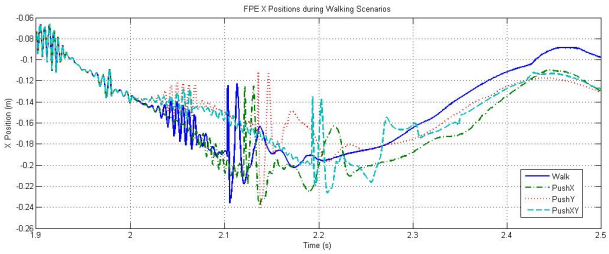
VI. Conclusions

This paper proposed an approach for on-line gait generation capable of responding to unknown disturbances. In the proposed approach, the point on the ground where the robot must step in order to regain stability following a disturbance is found via a two step process. First, the plane perpendicular

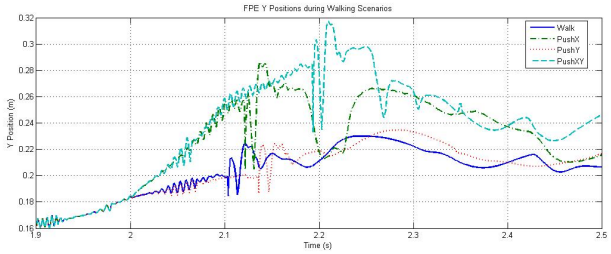


(a) Initial Pose (b) After Lean/Push states (c) After Lift state (d) After Swing state (e) In Stand state

Fig. 6. Simulation results when the robot tries to walk by leaning forward until it enters an unstable state, then recovering.



(a) X Position of FPE



(b) Y Position of FPE

Fig. 7. Comparison of FPE positions for the four different walking scenarios. The Swing state starts at 1.94 seconds, but the robot enters the Drop state at different times depending on the scenario. This transition from the Swing state to the Drop state is the cause of the large oscillations in FPE position between 2.1 and 2.2 seconds, as the controller switches from global to local reference frames to facilitate contact dynamics with the dropping foot.

to the overall angular momentum of the robot with respect to the ground projection of the COM is found, and then the location where the total energy of the biped after swing foot impact is equal to the peak potential energy within the plane is found as the target stepping point. This target is re-computed at each timestep, allowing the robot to adapt to disturbances on-line. A simple state machine is used to generate a full gait cycle; the robot initiates lifting of the swing leg either when forward progress is desired or when a disturbance is observed, and tracks the target foot placement location to determine the swing leg placement. The proposed approach is tested in simulation and shown to generate stabilizing foot placements to disturbances from arbitrary directions, both while standing still and in motion.

In future work, the proposed approach will be implemented and verified on a physical robot currently in construction at the University of Waterloo. The proposed approach will also be verified with uneven terrain, timing variations of the push during the gait cycle, and to determine the range and spatial distribution of disturbances that can be handled.

REFERENCES

[1] M. Vukobratović and B. Borovac, "Zero-moment point - thirty five years of its life," *International Journal of Humanoid Robotics*, vol. 1, no. 01, pp. 157–173, 2004.

[2] Q. Huang, K. Yokoi, S. Kajita, K. Kaneko, H. Arai, N. Koyachi, and K. Tanie, "Planning walking patterns for a biped robot," *Robotics and Automation, IEEE Transactions on*, vol. 17, no. 3, pp. 280–289, 2001.

[3] T. Takenaka, T. Matsumoto, and T. Yoshiike, "Real time motion generation and control for biped robot-1¡ sup¿ st¡/sup¿ report: Walking gait pattern generation," in *Intelligent Robots and Systems (IROS), 2009 IEEE-RSJ International Conference on*, 2009, pp. 1084–1091.

[4] J. Urata, K. Nshiwaki, Y. Nakanishi, K. Okada, S. Kagami, and M. Inaba, "Online decision of foot placement using singular lq preview regulation," in *Humanoid Robots, 2011 IEEE-RAS International Conference on*, 2011, pp. 13–18.

[5] A. Goswami, "Postural stability of biped robots and the foot-rotation indicator (fri) point," *The International Journal of Robotics Research*, vol. 18, no. 6, pp. 523–533, 1999.

[6] J. Pratt, J. Carff, S. Drakunov, and A. Goswami, "Capture point: A step toward humanoid push recovery," in *Humanoid Robots, 2006 IEEE-RAS International Conference on*, 2006, pp. 200–207.

[7] T. Koolen, T. De Boer, J. Rebula, A. Goswami, and J. Pratt, "Capturability-based analysis and control of legged locomotion, part 1: Theory and application to three simple gait models," *The International Journal of Robotics Research*, vol. 31, no. 9, pp. 1094–1113, 2012.

[8] J. Pratt, T. Koolen, T. De Boer, J. Rebula, S. Cotton, J. Carff, M. Johnson, and P. Neuhaus, "Capturability-based analysis and control of legged locomotion, part 2: Application to m2v2, a lower-body humanoid," *The International Journal of Robotics Research*, vol. 31, no. 10, pp. 1117–1133, 2012.

[9] B.-K. Cho, J.-H. Kim, and J.-H. Oh, "Balancing strategy using the principle of energy conservation for a hopping humanoid robot," *International Journal of Humanoid Robotics*, vol. 10, no. 03, 2013.

[10] D. L. Wight, E. G. Kubica, and D. W. Wang, "Introduction of the foot placement estimator: A dynamic measure of balance for bipedal robotics," *J. Comput. Nonlinear Dynam.*, vol. 3, no. 1, 2008.

[11] P. van Zutven, D. Kostic, and H. Nijmeijer, "Foot placement for planar bipeds with point feet," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, 2012, pp. 983–988.

[12] S. Choudhury and D. Kulić, "Gait generation via the foot placement estimator for 3d bipedal robots," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, 2013, pp. 5689–5695.

[13] M. Millard, J. McPhee, and E. Kubica, "Foot placement and balance in 3d," *J. Comput. Nonlinear Dynam.*, vol. 7, no. 2, 2012.

[14] S.-k. Yun and A. Goswami, "Momentum-based reactive stepping controller on level and non-level ground for humanoid robot push recovery," in *Intelligent Robots and Systems (IROS), 2011 IEEE-RSJ International Conference on*, 2011, pp. 3943–3950.

[15] S. Kajita, F. Kanehiro, K. Kaneko, K. Yokoi, and H. Hirukawa, "The 3d linear inverted pendulum mode: A simple modeling for a biped walking pattern generation," in *Intelligent Robots and Systems (IROS), 2001 IEEE-RSJ International Conference on*, 2001, pp. 239–246.

[16] S. Choudhury, D. Wight, and D. Kulić, "Rapid prototyping toolchain for humanoid robotics applications," in *Humanoid Robots, 2012 IEEE-RAS International Conference on*, 2012, pp. 817–822.