Trajectory Planning of a Bio-inspired Walker in 3D Cluttered Environments using Internal Models

Nicolas Van der Noot[1,2], Auke Jan Ijspeert[2] and Renaud Ronsse[1]

***Abstract*— Navigation of humanoids in cluttered environments is a complex task which requires sensorimotor coordination while maintaining the balance of the walker. Typically, robots rely on greedy computation of slow, inefficient and unnatural gaits. This contrasts with the relative ease and efficiency characterizing motion planning and execution of humans. In previous contributions, we developed a bio-inspired torque-based controller recruiting virtual muscles driven by reflexes and a central pattern generator. Speed control and steering could be achieved by the modulation of the forward speed and heading. This paper extends this controller to automatically compute both of these inputs, in order to achieve trajectory planning in 3D cluttered environments. To do so, we first develop a method based on internal models, a concept widespread in cognitive neuroscience. We then compare the obtained gait to results generated with a more traditional planning method based on potential fields. In particular, we show that internal models result in more robust gaits by taking the walker dynamics into account.**

I. INTRODUCTION

Humanoid robots hold the promise of a wider spread of robotics use in our everyday lives as their body is very similar to the human one, and thus more adapted to move in environments designed for humans [1]. However, their locomotion in uncontrolled environments remains an issue, as emphasized by the DARPA Robotics Challenge [2]. On top of their poor robustness in cluttered environments, typical humanoid gait controllers — such as those recruiting the zero-moment point as an indicator of gait feasibility [3] — are usually energetically inefficient and exhibit unnatural gaits at slow speeds [4], [5].

In contrast, humans are capable of robust, efficient and versatile locomotion in diverse and cluttered environments. Aiming at capturing some of these typical human walking features, the seminal work of Geyer and Herr reported the development of a bipedal model actuated by a human-like neuromuscular model recruiting reflex signals [6]. We incremented this model with the inclusion of central pattern generators [7] to obtain modulation capabilities, first in 2D [8], and later in 3D environments [9]. Finally, we extended it to control the robot speed and steering (direction and curvature) by the modulation of two simple scalar inputs: the forward speed and heading references [10].

Yet, the modulation of both of these inputs needed to be performed by a human operator. In particular, moving in a cluttered environment was not as straightforward as moving over a flat ground because the curvature was intrinsically limited and because it was not possible to plan the exact foot placement. In [11], we proposed an approach to modulate the step height and length, but this required additional optimizations and did not provide a direct way to deal with obstacles. To address these shortcomings, this paper develops a higher-level controller in charge of computing the forward speed and heading references, in order to achieve trajectory planning in cluttered environments.

The proposed approach recruits internal models of the robot neuro-musculo-skeletal apparatus. Neural internal models are circuitries that can learn the input/output relationships of the motor apparatus [12]. In particular, the cerebellum is thought to perform system identification by building internal models that predict sensory outcome of motor commands and correct motor commands through internal feedback [13].

Systems modeling the causal relationship between actions and their consequences are known as forward internal models. They anticipate the interplay between the body and the world in order to overcome time delays associated with sensory feedback control [14], [15]. Inverse internal models, on the other hand, can compute feed-forward motor commands from desired trajectory information [12]. They are therefore well suited to act as controllers as they can provide the motor command necessary to achieve some desired state transitions [14]. In theory, a forward model of the motor apparatus embedded in an internal feedback loop can approximate an inverse model [12].

On top of embedding the robot with an internal model of its neural system, we coupled it to a biomechanical model of its body evolving in the locomotion environment. Therefore, such a model can be casted as a digital twin of the robot. Digital twins are digital replications of living as well as non-living physical entities, enabling data to be seamlessly transmitted between the physical and virtual worlds in order to monitor, understand, and optimize the physical entity behavior [16]. Here, these digital twins are used to anticipate the consequences of a given set of commands to navigate in cluttered environments, and therefore optimize these commands for trajectory planning.

Finally, the trajectory and gait generated using internal models will be compared to those obtained with a more

This research was supported by the European Community's Seventh Framework Programme under Grant 611832 (WALK-MAN) and by the Belgian F.R.S.-FNRS (Aspirant #16744574 awarded to NVdN).

[1]N. Van der Noot and R. Ronsse are with the Institute of Mechanics, Materials and Civil Engineering; the Institute of Neuroscience; and "Louvain Bionics", Université catholique de Louvain, B-1348 Louvain-la-Neuve, Belgium. renaud.ronsse@uclouvain.be

[2]N. Van der Noot and A. J. Ijspeert are with the Biorobotics Laboratory, Institute of Bioengineering, École Polytechnique Fédérale de Lausanne, CH-1015 Lausanne, Switzerland. auke.ispeert@epfl.ch

traditional localization and navigation technique, historically developed for wheeled robots and ported to humanoids, namely the potential fields [17].

This paper is organized as follows. In Section II, we review the simulated robotic platform used in this research and detail the controller main parameters. These parameters are later obtained using internal models in Section III and potential fields in Section IV. Section V then compares the trajectories and gaits obtained using both of these methods. Finally, Section VI concludes the paper.

II. MODEL OF STEERABLE BIPED

This paper uses the COMAN platform, a 95 cm tall humanoid robot, as embodiment. As reported in [9] and [10], we use the Robotran simulator to model the robot in its environment. The robot's 23 joints are torque-controlled; these torque references come from virtual muscles, fed by virtual reflexes and a central pattern generator. More information about the robot and its model can be found in the aforementioned papers and in [18].

Although all the developments of this contribution are done in simulation, the final purpose is to port these algorithms on a real platform. Therefore, we only use inputs available to the actual robot (or which could be obtained with realistic additions) and we introduce noise on the torque readings, similarly to what would happen on the real platform. In this paper, we chose to refer to the simulated robot as the "real robot" to distinguish it from its digital twins (see Section III). Finally, we assume that the robot knows its environment (obstacles positions...), a feature that could be provided to the real platform by using a vision system.

In [9] and [10], we developed a bio-inspired controller capable of generating human-like gaits with steering capabilities. More precisely, two high-level inputs, namely the speed (i.e. v_{ref}) and heading (i.e. h_{ref}) references, could be adapted on-line by an external operator, resulting in speed and steering, i.e. path and curvature, adaptations.

Yet, the exact feet placement was difficult to predict. This had a significant impact when the robot needed to avoid small obstacles and/or holes along its path. Since the steering curvature was limited, there was no straightforward method for an external operator to adapt the commands to reach a target, while avoiding obstacles.

In this contribution, we present two different methods to automatically track a moving goal position, while avoiding obstacles. The first method is based on the development of internal models, while the second one recruits a potential field navigation algorithm. In other words, this contribution develops a higher layer to our pre-existing *reference controller* (developed in [9] and [10]), in order to automatically compute v_{ref} and h_{ref} .

III. INTERNAL MODELS PREDICTION

In order to develop internal models capable of predicting the future motion of the robot in cluttered environments, we designed the following strategy. Seven digital twins of the real robot simulated potential trajectories in its current environment. These models were running in parallel four times faster than the real robot to serve as predictors of potential locomotion trajectories. Each model received different sets of speed and heading command to test. After a short time, the robot selected the command corresponding to the most robust and efficient model. This process was repeated up to the end of the task, i.e. when the robot reached its target position. These internal models were therefore able to predict the future step locations of the robot and to check if the corresponding path was safe.

A. Internal models state

In order to generate internal models from the whole body dynamics, a compact representation with low dimensionality can generally be extracted, known as the state of the system. This state contains all the relevant time-varying information needed to predict or control the future of the system [15]. In humans, the cerebellum is assumed to be involved in state estimation through monitoring of efferent copies [19].

Here, we identified 172 state variables available to the controller, which had to be provided to the internal models to rebuild the current state of the real robot. On top of that, each internal model received the current position (including the absolute orientation) and speed of the real robot waist. This represents 12 additional state variables, which are currently not available with the real platform. Yet, they could later be obtained on the real hardware by developing specific localization hardware and software, or possibly by relying on external measurement devices. Every second, a reset of these internal models was performed in order to remain close to the current state of the robot, and to incorporate its changes in speed and heading command.

The 172 controller state variables are the following (the number of floating parameters is indicated in bold; they are all described in [9] and [10]):

- position and velocity of the 23 motors actuating the robot's degrees of freedom, i.e. before the series elastic actuators (SEA), see [20] (**46**)
- position and velocity of the 23 corresponding joints, i.e. after the SEA (**46**)
- time (**1**)
- state of each leg: swing or stance (**2**)
- supporting leg state (see [9]) (**1**)
- length of the contractile element of the 48 virtual muscles actuating the robot's joints (**48**)
- state of the oscillators being the building units of the central pattern generator (**24**)
- state of both neurons providing turning commands (**2**)
- speed (v_{ref}) and heading (h_{ref}) commands (**2**)

With the addition of the 12 kinematic variables of the waist, there are thus 184 parameters to be sent to the internal models in order to synchronize them with the current state of the real robot. Importantly, the accelerations of the joints, motors and of the waist position are not provided (and therefore set to zero). While these measures could be easily obtained in simulation, their accurate measurement on

the real robot is unrealistic. This fact combined with the noise introduced in the torque sensory readings resulted in small differences between the internal models predictions and the future motion of the robot, as would happen with real hardware.

In contrast to [10], all the steering adaptations (related to the heading reference) were applied in the phase following the last foot touch down of either leg. This was done in order to obtain faster heading changes, at the cost of a reduced lateral stability.

Also, synchronizations were never performed during the double support phase. If a synchronization was triggered during a double support phase, it was performed at the next swing phase. The underlying reason is because the ground contact model (GCM) is more sensitive to numerical issues when both feet are in contact with the ground, due to the state machine used for the GCM (see [9]).

B. Internal models to avoid obstacles

The first validation task required the robot to walk in a 3D simulation environment with obstacles and walls that must be avoided. It is called "Experiment 1" and is shown in the first part of the multimedia attachment (video *internal_models.mp4*). In the next figures, the real robot is represented with a blue torso, while its internal models have other colors for the torso. Therefore, the real robot is referred to as the "blue robot". Fig. 1 displays the blue robot in its environment, and the orange internal model predicting the future motion of the blue robot.

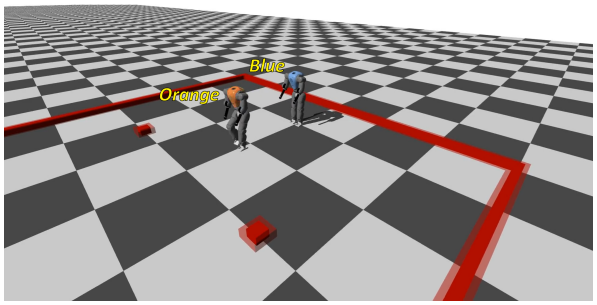


Fig. 1. The COMAN robot (blue torso) must navigate in a cluttered environment (red obstacles and walls). Internal models (like the one with the orange torso) run four times faster than the COMAN robot in order to forecast its future motion. The internal models walk in the same environment as the blue robot, except for obstacles that are widened by 10 cm (semi-transparent red shapes).

As detailed in Section III-A, the internal models do not perfectly predict the future motion of the robot, due to noise and to the lack of information about accelerations. In order to compensate for these inaccuracies, the obstacles are widened by an arbitrary safety margin of 10 cm (i.e. each edge is inflated by 5 cm), see Fig. 1. This holds only for the internal models and not for the "real" robot.

The internal models are initially launched while the blue robot is still in the upright position (see Fig. 1). After the four initial steps of the blue robot (i.e. corresponding to steady state walking), the internal models are re-synchronized with this blue robot, as detailed in Section III-A. Subsequently, all synchronizations are performed every second.

TABLE I

INTERNAL MODELS COMMANDS. HEADING REFERENCES: $h_{ref} = 0$ CORRESPONDS TO STRAIGHT WALKING, $h_{ref} < 0$ TO LEFT TURNING, AND $h_{ref} > 0$ TO RIGHT TURNING.

ID	color	$v_{ref}[m/s]$	$h_{ref}[-]$
1	red	0.65	0
2	green	0.5	0
3	purple	0.8	0
4	yellow	0.55	0.8
5	orange	0.55	-0.8
6	white	0.75	0.8
7	black	0.75	-0.8

Each internal model tests the impact of a given set of v_{ref} and h_{ref} commands on the robot path. The commands received by each internal model are detailed in Table I. They result in different paths and steps for the different models, as illustrated in Fig. 2.

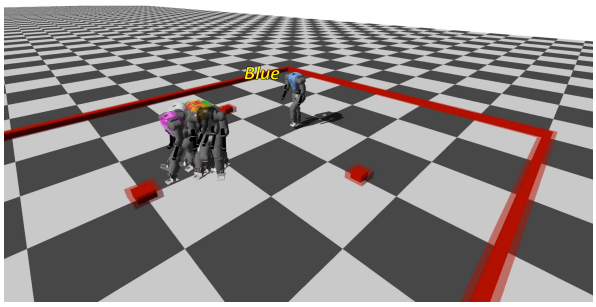


Fig. 2. The seven internal models receive different speed and heading commands, resulting in different paths and steps.

One second after resetting their states, each internal model is expected to have simulated four seconds and to evaluate its performance. This evaluation is based on the simulated walking time of the internal model before falling (this time is considered as infinite if the model did not fall). Subsequently, the blue robot gathers all these scores and adapts its next commands according to the ones selected by its most promising internal model (see Fig. 3). If the seven internal models fell, the robot selects the internal model with the longest walking time before falling. Otherwise, a model is arbitrarily picked among the surviving ones.

Because the internal models require one second of real time to evaluate their performance, they initially keep the same commands as the blue robot when predicting the first second of the blue robot future motion. Then, they linearly adapt their commands to their target command. This linear adaptation is performed in 0.5 second of predicted time. After one second of real time (thus corresponding to four seconds of predicted time), the blue robot linearly adapts its current commands to the ones selected by its best internal model, with a linear adaptation lasting 0.5 second (of real time).

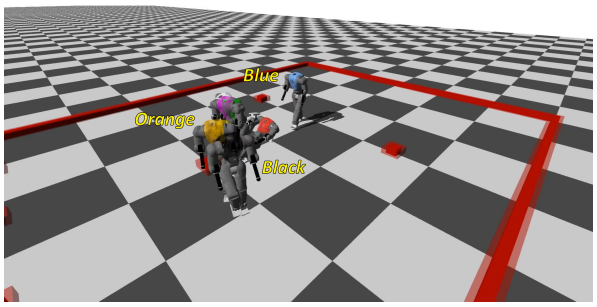


Fig. 3. The blue robot adapts its commands according to the ones selected by its most promising internal models. In this case, the orange and the black models did not fall. The blue robot will thus select the commands of one of these two models.

In sum, after re-synchronizing its seven internal models, the blue robot keeps its previous commands during 1 second. In the meantime, the internal models predict what will be the state of the robot in the next 4 seconds, when using seven different sets of commands. At the end of this process (lasting 1 second of real time), the blue robot selects the best commands evaluated by its internal models, and linearly adapts its commands during 0.5 second. Subsequently, the internal models are reset and synchronized with the blue robot as soon as the blue robot is not in double support phase, and this cycle is repeated up to the end of the task.

C. Target reaching in cluttered environments using internal models

The strategy presented in Section III-B can be adapted to incorporate a moving target (”Experiment 2”, shown in the second part of the multimedia attachment). The position of the target is controlled in real-time by an external operator. In the experiment displayed in Fig. 4, the target (depicted by the blue arrow) was controlled using a joystick. The robot purpose was to adapt its commands to come close to the target (called ”goal position”), while avoiding the obstacles.

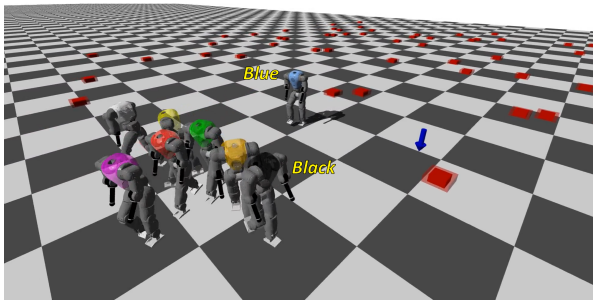


Fig. 4. The goal position is controlled by the external operator and depicted by the blue arrow. In this case, the black internal model is the most likely to reach this target.

The strategy of Section III-B is kept intact, except for the evaluation of the best commands. If all the internal models fell at the end of the 4 seconds, the one walking the longer time before falling is selected. However, if one or more models survive by the end of these 4 seconds, the following score S is computed among the surviving models: $S = k_d\, ln(1 + \Delta d) + k_\alpha\, \Delta\alpha$, where k_d and k_α are two constant respectively set to 1 and 2 (arbitrary values), Δd is the distance between the goal and the model final position (i.e. at the end of the 4 seconds) and $\Delta\alpha$ is the absolute value of the angle between the robot final orientation, and the oriented line from its final position to the goal, i.e. absolute value of the angle error in the transverse plane. The model with the lowest score S is selected.

This selection can be interpreted as following: Δd and $\Delta\alpha$ should be minimized in order to be more likely to reach the final target. k_d and k_α are scaling factors. The operator $ln(1 + \Delta d)$ is used to reduce the importance of a faraway target, while keeping a positive value, even when $\Delta d = 0$. In the example of Fig. 4, the black robot has the lowest score S value, and is therefore selected.

IV. POTENTIAL FIELD PLANNING

The second method to compute appropriate (v_{ref}, h_{ref}) is based on the well-known potential field navigation algorithm [21]. This method is intensively used in mobile robotics because it achieves robust goal tracking and obstacle avoidance, at a low computational cost. We directly report the solution to handle both obstacles and a target to be reached (i.e. corresponding to ”Experiment 2” and to Section III-C).

A known drawback of potential field path planning is that it requires a lot of parameters hand tuning. Here, we thus conducted several preliminary tests leading to the parameters tuning reported below, that was identified as a good compromise between robustness and performance (here, target tracking).

A. Potential fields computation

The purpose of this potential field module is to compute a force $\mathbf{F} = (F_x, F_y)$ driving the robot from its current position (x_r, y_r) (respectively along the X and Y axes) to the goal position (x_g, y_g), while avoiding obstacles.

Two different contributions are summed:

- one attractive potential field related to the goal position
- N repulsive potential fields related to each obstacle

First, the attractive potential field is parabolic and computed as $U_{att} = 0.5\, k_{att}\, (\Delta x^2 + \Delta y^2)$, where $\Delta x = x_r - x_g$, $\Delta y = y_r - y_g$ and the proportional gain k_{att} is set to 2.

The resulting force along the X axis $F_{x,att}$ is computed as follows: $-\partial U_{att}/\partial x_r = -k_{att}\, (x_r - x_g)$, and a similar equation is used for the one along the Y axis $F_{y,att}$.

The amplitude of the resulting force (i.e. $||\mathbf{F}|| = \sqrt{F_x^2 + F_y^2}$) is bounded to an upper limit of 10.

Then, the total repulsive potential field U_{rep} is computed as the sum of the repulsive fields $U_{rep,i}$ of each obstacle i (i.e. $U_{rep} = \sum_i^N U_{rep,i}$; $F_{rep} = \sum_i^N F_{rep,i}$). In the following equations, ρ_i is the shortest distance between the robot body and the obstacle i (in [m]), while ρ_0 (set to 4 m) is the distance of influence of each obstacle. The repulsive potential of each obstacle i is only computed when the robot lies within this distance of influence.

For $0 < \rho_i < \rho_0$, the repulsive potential field of obstacle i is computed as $U_{rep,i} = 0.5\, k_{rep}\, (1/\rho_i - 1/\rho_0)^2$, where k_{rep} is the repulsive constant, set to 1.

The corresponding repulsive force along the X axis $F_{x,rep,i}$ is computed as follows: $-\partial U_{rep,i}/\partial x_r = k_{rep}\,(\frac{1}{\rho_i} - \frac{1}{\rho_0})\,\frac{\delta x}{\rho_i^3}$ where δx is the distance from the robot position to the closest obstacle point along the X axis. A similar equation is used for the repulsive force along the Y axis $F_{y,rep,i}$.

All these potential fields and forces are summed. Finally, the resulting total force amplitude is bounded to an upper value of 100. An example of potential fields map is visible in Fig. 5, corresponding to the obstacles visible in the second part of the multimedia attachment ("Experiment 2"). Here, the goal location is set to $(x_g, y_g) = (8, 0)$ and the potential fields are computed for all the points of the map. This is only for the sake of visualization because only the potential fields force at the robot's position (i.e. (x_r, y_r)) needs to be computed in this approach.

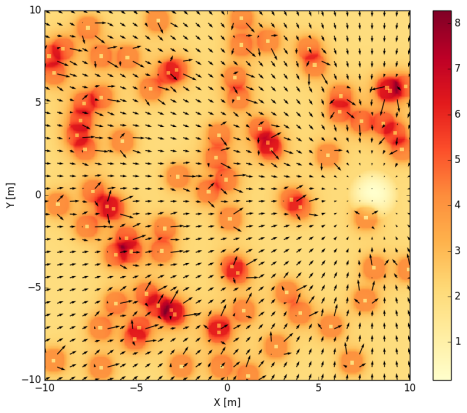


Fig. 5. Visualization of the potential field with the obstacles of "Experiment 2" and a goal point set to $(8, 0)$.The resulting potential field U is depicted with colors corresponding to the ones of the color bar (with U_{att} and each $U_{rep,i}$ bounded to an upper limit of 2). The resulting forces are represented by the black arrows.

B. Potential field walking commands

Section IV-A computed the resulting force $\mathbf{F} = (F_x, F_y)$ at the robot position (x_r, y_r). Here, we use it to obtain appropriate commands (v_{ref}, h_{ref}) for the robot. In the following equations, $\theta_F = \text{atan2}(F_y, F_x)$ is the orientation of the force vector and θ_R is the robot's orientation. Each orientation is defined by its corresponding angle in the transverse plane, bounded in the $[-\pi; \pi]$ interval, and with a value of 0 rad corresponding to an orientation aligned with the positive X axis.

First, the angle $\Delta\theta = \theta_R - \theta_F$ is computed and translated in the $[-\pi; \pi]$ interval. The heading reference h_{ref} is then computed as $\frac{\Delta\theta}{\pi/2}$, in case $-\pi/2 \leq \Delta\theta \leq \pi/2$. Otherwise, h_{ref} is set to -1 if $\Delta\theta < -\pi/2$, while h_{ref} is set to 1 if $\Delta\theta > \pi/2$. This makes the robot progressively aligning its orientation towards the target point (with a correction proportional to the orientation error).

The speed reference v_{ref} is set to $v_{MIN} + k_v\,(v_{MAX} - v_{MIN})$, where v_{MIN} and v_{MAX} are respectively the minimal (0.4 m/s) and maximal values (0.9 m/s) of the speed reference. k_v is a variable gain bounded in the $[0; 1]$ interval and is computed as $[||\mathbf{F}||/10]_{max=1}\,[\cos(\Delta\theta)]^+$, where $[\bullet]_{max=1}$ saturates its argument to 1 and $[\bullet]^+ = max(0, \bullet)$. In this way, the robot moves faster when the force amplitude $||\mathbf{F}||$ is large and slows down for small values of $||\mathbf{F}||$. Small values of $||\mathbf{F}||$ usually happen when the robot is close to its target or when it is close to obstacles with force fields opposite to the target direction. On top of that, the function $[\cos(\Delta\theta)]^+$ reduces the speed coefficient k_v when the robot is not properly aligned with the force field, thus decreasing its speed to have more time in order to correct its trajectory.

Interestingly, this computation method allows the commands (v_{ref}, h_{ref}) to be continuously adapted (i.e. no commands discontinuity), provided there is no discontinuity in the goal and obstacles positions.

V. RESULTS

Results from Experiments 1 and 2 are reported in the multimedia attachment (video *internal_models.mp4*). Results reported below are obtained while the robot is requested to follow a moving goal, either in an empty or in a cluttered environment.

A. Tracking of a time-varying trajectory

Here, the goal position is updated so that the robot needs to switch between straight lines, and right and left turning motions. We arbitrarily decided to describe the goal position motion following the ∞ symbol, mathematically casted as a lemniscate of Bernoulli.

The goal position is initially set to $(x_{g,0}, y_{g,0}) = (5, 0)$. After 5 s, this position is updated as follows. First, a scaled time t' is computed as $2\,\pi\,\frac{t-5}{\Delta T_{lem}} - \pi/2$, where t is the real time (in s), and ΔT_{lem} is the period to have a full cycle of the lemniscate. The time-varying goal position (x_g, y_g) is then computed as follows:

$$\begin{aligned} x_g &= x_{g,0} + \frac{A_{lem}\,\sqrt{2}\,\cos(t')}{\sin^2(t') + 1} \\ y_g &= y_{g,0} + \frac{A_{lem}\,\sqrt{2}\,\cos(t')\,\sin(t')}{\sin^2(t') + 1} \end{aligned} \tag{1}$$

where A_{lem} is set to 10, and was arbitrarily selected to have curves with steering radii being achievable by the robot (see [10] for more details).

In Fig. 6, this scenario was tested in a flat and empty environment (i.e. no obstacle, no hole). The target was moving during 320 s using $\Delta T_{lem} = 100$ s. The paths of the robot were measured using both internal models and potential fields. Fig. 6 shows that the method based on the potential fields managed to stay closer to the reference path. This is mainly due to the discretization of commands used by the internal models approach (i.e. only 7 discrete sets of commands being evaluated), while the potential fields feature a continuous command modulation.

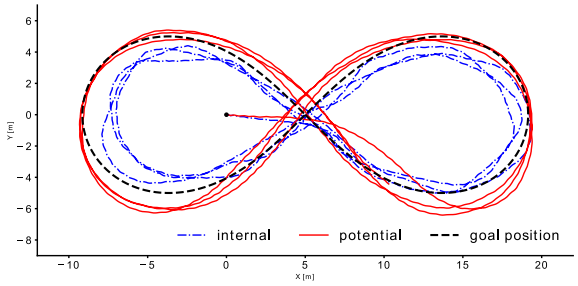


Fig. 6. The goal position is depicted with dashed lines and is updated as detailed in Eq. (1), starting at $(x_{g,0}, y_{g,0}) = (5, 0)$ and using $\Delta T_{lem} = 100\,\mathrm{s}$. COMAN was initially located at $(x_{r,0}, y_{r,0}) = (0, 0)$, pointing towards the positive X axis. The robot path obtained with the internal models and potential field method are respectively represented in blue and red. The full experiment lasted 320 s, so that the target made a bit more than three full cycles.

B. Robustness in cluttered environments

Here, we keep the same trajectory as the one described by Eq. (1) and visible in Fig. 6. However, the environment is cluttered with randomly placed obstacles or holes.

In a first scenario (i.e. ground with obstacles), each obstacle is a box of $20\,\mathrm{cm} \times 20\,\mathrm{cm}$, aligned with the X and Y axes, with a height of 10 cm, i.e. too high to step over it. They are similar to the ones visible in "Experiment 2" (see multimedia attachment) and in Fig. 5. These obstacles are randomly placed (flat distribution) on the area with the following boundaries: x bounded between -15 m and 25 m; y bounded between -10 m and 10 m. This represents a surface of $800\,m^2$. With n_{obs} being the number of obstacles, this represents an average obstacles density of $n_{obs}/800$ obstacles per square meter. Importantly, no obstacle is placed in a radius of 1 m around the robot starting position (see Fig. 6), and in a safety rectangle of $4\,\mathrm{m}$ (along the X axis) $\times\, 2\,\mathrm{m}$ (along the Y axis) right in front of the robot. This prevents the robot from starting from an impossible configuration where no path could be found.

The second scenario (i.e. ground with holes) is similar to the first one, except that all the obstacles are replaced by holes of the same dimension, i.e. squares of $20\,\mathrm{cm} \times 20\,\mathrm{cm}$ with a depth of 10 cm.

These two scenarios are tested for the following number of obstacles n_{obs}: 0, 50, 100, 150, 200, 250, 300 and for the following durations ΔT_{lem}: 90, 100, 110, 120, 130, 140 s. These durations respectively correspond to average target speeds of 0.82, 0.74, 0.67, 0.62, 0.57, 0.53 m/s (i.e. to speeds in the speed range capabilities of the robot). The resulting robustness of the two methods (i.e. internal models and potential fields) is evaluated as the average time the robot walks in these environments while following the target, before falling.

As illustrated in Fig. 7, the robustness of the first scenario (i.e. ground with obstacles) is slightly better for the internal models method, especially for $n_{obs} \leq 50$. However, this difference is not significant. Regarding the second scenario (i.e. ground with holes), the results are also globally better for the internal models method, but the differences are more significant, especially for $\Delta T_{lem} \leq 110\,\mathrm{s}$ and so for average target speeds larger than 0.67 m/s. In other words, internal models prediction performs better for speeds around the middle of the COMAN's speed range, and faster.

VI. CONCLUSION

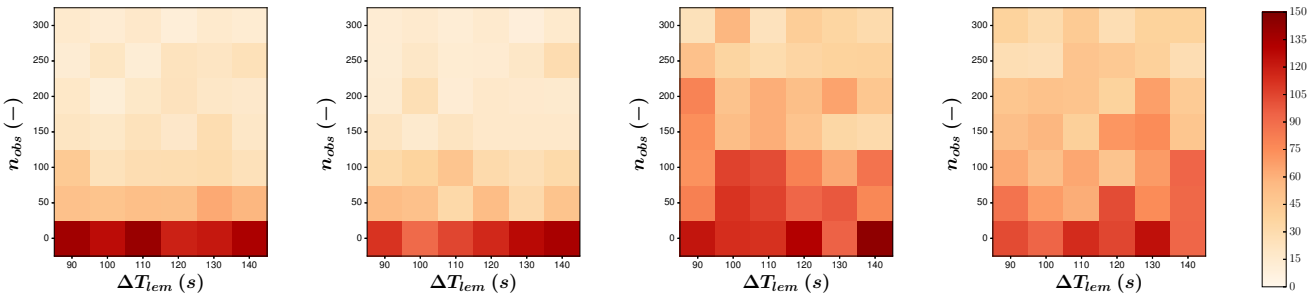
In this paper, we presented a method recruiting internal models to predict the future steps of a bipedal robot and to adapt its path in cluttered environments. We compared it to a more traditional controller based on potential fields and showed that both approaches have their respective pros and cons.

Indeed, gaits generated with the internal models show an increased robustness compared to the gaits obtained with the potential field method. The internal models notably take the full dynamics into account to predict the impact of obstacles and holes. This is particularly relevant to cross holes without considering them as impassable obstacles. Other advantages include the small amount of parameters to tune (only the synchronization period, the commands to evaluate and the score) and the lack of local minimum problem, typical of potential field methods.

In contrast, gaits obtained with the potential field approach managed to stay closer to the reference path in unobstructed flat terrains, with a much smaller computational cost as compared to the internal models. Regarding this last point, a future perspective of the current contribution could be to simplify the internal models without modeling the full body dynamics. In [22], a simple forward internal model, recruiting a three-layer feed-forward neural network, was used to predict the expected self-generated acceleration during walking, in turn stabilizing walking on changing slopes.

Another perspective of this work would be to use more than seven internal models, in order to test a larger set of commands, and therefore a better motion granularity. The use of internal models could also be tested on the real platform, thanks to the transfer framework we developed in [23]. In this contribution, we simulated the "real robot" by using a similar model as the ones used for the digital twins. Adapting it for the real world would thus probably require the introduction of a more accurate noise modeling in the internal models. On a real platform, our framework could further be combined with an online mechanism correcting the accuracy of the dynamic model [24], in order to improve the internal model predictability, and thus the quality of planning.

Finally, the internal models could potentially also be used in a synchronous mode. In other words, an internal model could run at the same speed as the real robot, in order to detect perturbations. For instance, the difference between the center of mass (COM) of the real robot and its internal model could be monitored. Then, if a push was applied to the real robot, this would cause the COM of the robot and its internal model to diverge. This difference could then be used by a controller like the one described in [25], in order to add reflexes mitigating this perturbation.



(a) obstacles, int. models (b) obstacles, pot. field (c) holes, int. models (d) holes, pot. field

Fig. 7. Measures of the walker's robustness in the two scenarios described in Section V-B while following a moving goal position (see Section V-A). For each set of number of obstacles n_{obs} and track duration ΔT_{lem}, the walker has to walk while following the goal. Its performance is evaluated as the period of time between the target initial motion (i.e. 5 s after the simulation initialization) and the moment the robot felt, limited to an upper bound of 150 s. This experiment is repeated ten times in a row, each time with new random obstacles placement. The color map represents the average performance (i.e. the aforementioned period of time) over the ten runs. Panels (a) and (b) gather the results of the first scenario (i.e. ground with obstacles), while panels (c) and (d) gather the results of the second scenario (i.e. ground with holes). Finally, panels (a) and (c) use the internal models method, while panels (b) and (d) use the potential field method.

REFERENCES

[1] S. Schaal, "The New Robotics-towards human-centered machines." *HFSP journal*, vol. 1, no. 2, pp. 115–26, July 2007.

[2] M. Johnson, B. Shrewsbury, S. Bertrand, D. Calvert, T. Wu, D. Duran, D. Stephen, N. Mertins, J. Carff, W. Rifenburgh, J. Smith, C. Schmidt-Wetekam, D. Faconti, A. Graber-Tilton, N. Eyssette, T. Meier, I. Kalkov, T. Craig, N. Payton, S. McCrory, G. Wiedebach, B. Layton, P. Neuhaus, and J. Pratt, "Team IHMC's Lessons Learned from the DARPA Robotics Challenge: Finding Data in the Rubble," *Journal of Field Robotics*, Sept. 2016.

[3] M. Vukobratovic 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.

[4] R. Kurazume, S. Tanaka, M. Yamashita, T. Hasegawa, and K. Yoneda, "Straight legged walking of a biped robot," in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2005, pp. 337–343.

[5] H. Dallali, "Modelling and dynamic stabilization of a compliant humanoid robot, CoMan," Ph.D. dissertation, University of Manchester, 2011.

[6] H. Geyer and H. Herr, "A muscle-reflex model that encodes principles of legged mechanics produces human walking dynamics and muscle activities." *IEEE transactions on neural systems and rehabilitation engineering : a publication of the IEEE Engineering in Medicine and Biology Society*, vol. 18, no. 3, pp. 263–73, June 2010.

[7] 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.

[8] N. Van der Noot, A. J. Ijspeert, and R. Ronsse, "Biped gait controller for large speed variations, combining reflexes and a central pattern generator in a neuromuscular model," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, Seattle, WA, May 2015, pp. 6267–6274.

[9] ——, "Bio-inspired controller achieving forward speed modulation with a 3D bipedal walker," *The International Journal of Robotics Research*, vol. 37, no. 1, pp. 168–196, Jan. 2018.

[10] N. Van der Noot, A. J. Ijspeert, and R. Ronsse, "Neuromuscular model achieving speed control and steering with a 3D bipedal walker," *Autonomous Robots*, vol. 43, no. 6, pp. 1537–1554, Aug. 2019.

[11] P. Greiner, N. Van der Noot, A. J. Ljspeert, and R. Ronsse, "Continuous Modulation of Step Height and Length in Bipedal Walking, Combining Reflexes and a Central Pattern Generator," in *2018 7th IEEE International Conference on Biomedical Robotics and Biomechatronics (Biorob)*, Aug. 2018, pp. 342–349, iSSN: 2155-1774.

[12] M. Kawato, "Internal models for motor control and trajectory planning," *Current Opinion in Neurobiology*, vol. 9, no. 6, pp. 718–727, Dec. 1999.

[13] R. Shadmehr and J. W. Krakauer, "A computational neuroanatomy for motor control," *Experimental Brain Research*, vol. 185, no. 3, pp. 359–381, Mar. 2008.

[14] D. M. Wolpert, R. C. Miall, and M. Kawato, "Internal models in the cerebellum," *Trends in Cognitive Sciences*, vol. 2, no. 9, pp. 338–347, Sept. 1998.

[15] D. M. Wolpert and Z. Ghahramani, "Computational principles of movement neuroscience," *Nature Neuroscience*, vol. 3, no. 11, pp. 1212–1217, Nov. 2000.

[16] A. El Saddik, "Digital Twins: The Convergence of Multimedia Technologies," *IEEE MultiMedia*, vol. 25, no. 2, pp. 87–92, Apr. 2018.

[17] M. Ferro, A. Paolillo, A. Cherubini, and M. Vendittelli, "Omnidirectional humanoid navigation in cluttered environments based on optical flow information," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, Nov. 2016, pp. 75–80, iSSN: 2164-0580.

[18] A. A. Zobova, T. Habra, N. Van der Noot, H. Dallali, N. G. Tsagarakis, P. Fisette, and R. Ronsse, "Multi-physics modelling of a compliant humanoid robot," *Multibody System Dynamics*, vol. 39, no. 1-2, pp. 95–114, Jan. 2017.

[19] J.-J. Orban de Xivry and V. Ethier, "Neural Correlates of Internal Models," *The Journal of Neuroscience*, vol. 28, no. 32, pp. 7931–7932, Aug. 2008.

[20] M. Mosadeghzad, G. a. Medrano-Cerda, J. a. Saglia, N. G. Tsagarakis, and D. G. Caldwell, "Comparison of various active impedance control approaches, modeling, implementation, passivity, stability and trade-offs," in *IEEE/ASME International Conference on Advanced Intelligent Mechatronics, AIM*. IEEE, July 2012, pp. 342–348.

[21] L. E. Kavraki and S. M. LaValle, "Motion Planning," in *Springer Handbook of Robotics*, B. Siciliano and O. Khatib, Eds. Berlin, Heidelberg: Springer, 2008, pp. 109–131.

[22] J. Schröder-Schetelig, P. Manoonpong, and F. Wörgötter, "Using efference copy and a forward internal model for adaptive biped walking," *Autonomous Robots*, vol. 29, no. 3, pp. 357–366, Nov. 2010.

[23] N. Van der Noot, L. Colasanto, A. Barrea, J. van den Kieboom, R. Ronsse, and A. J. Ijspeert, "Experimental validation of a bio-inspired controller for dynamic walking with a humanoid robot," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept. 2015, pp. 393–400.

[24] D. Nguyen-Tuong and J. Peters, "Model learning for robot control: a survey," *Cognitive Processing*, vol. 12, no. 4, pp. 319–340, Nov. 2011.

[25] F. Heremans, N. Van der Noot, A. J. Ijspeert, and R. Ronsse, "Bio-inspired balance controller for a humanoid robot," in *2016 6th IEEE International Conference on Biomedical Robotics and Biomechatronics (BioRob)*, June 2016, pp. 441–448.