Real-Time Stabilisation for Hexapod Robots

Marcus Hörger, Navinda Kottege, Tirthankar Bandyopadhyay, Alberto Elfes, and Peyman Moghadam

Autonomous Systems, CSIRO Computational Informatics,
Queensland Center for Advanced Technology, Brisbane, QLD 4069

Abstract. Legged robots such as hexapod robots are capable of navigating in rough and unstructured terrain. When the terrain model is either known *a priori* or is observed by on-board sensors, motion planners can be used to give desired motion and stability for the robot. However, unexpected leg disturbances could occur due to inaccuracies of the model or sensors or simply due to the dynamic nature of the terrain. We provide a state space based framework for stabilisation of a high dimensional multi-legged robot which detects and recovers from unexpected events such as leg slip. We experimentally evaluate our approach using a modified PhantomX hexapod robot with extended tibia segments which significantly reduces its stability. Our results show that roll and pitch stability is improved by 2× when using the proposed method.

1 Introduction

Legged robots such as hexapods shown in figure 1 are well suited for navigating in rough and uneven terrain that can be challenging to conventional wheeled or tracked vehicles. These robots can adapt to the complex terrain by adjusting their gait patterns, footfall trajectory or footholds. For effective and stable navigation in such terrain, the robot requires a mechanism to detect and recover from unexpected events such as leg slippage. In the event of a slip, identifying that the slip has occurred and subsequently taking corrective actions to move to a stable configuration allows it to continue its navigation task. Often such stabilisation requires the control of the whole system rather than just the individual leg that has slipped. This implies that a simple servo control on the leg motors alone are incapable of achieving the desired outcome.

In this paper, we frame the problem of hexapod stabilisation as a path finding problem from a critical region to a stable region in the configuration space of the robot. Such a formulation allows us to take advantage of the whole body kinematics in generating stable postures for future slip proof steps without explicit knowledge of the local terrain.

There are broadly two approaches in attaining hexapod locomotion: when terrain is unknown, the hexapod executes repeated pattern of coupled footfall or a gait with little feedback [1]; and when the terrain model is completely known or is observed by on-board sensors with sufficient accuracy, the footfalls are computed to give desired motion and stability [2] [3] [4] [5]. The former approach

(a) (b)

Fig. 1. Modified PhantomX hexapods (a) with additional computing and sensing, (b) with extended tibia segments which reduces its stability during locomotion.

relies on the stochastic nature of the hexapod's interaction with the terrain to recover from slips and trips. However, there is no guarantee of the approach working in very challenging environments like steep slopes or on slippery surfaces in the event of unexpected disturbances in the leg-terrain interaction such as leg slips or changes in the body orientation due to incomplete and uncertain terrain information. In order to navigate such challenging terrains, a fast reactive approach is necessary which is able to compensate these unforeseen disturbances. Another approach uses a set of behaviours to control a hexapod robot [6][7]. In order for a robot to adapt its behaviours, it must have the ability to autonomously detect and classify terrain types [8]. An issue which is not adequately addressed in such an approach is the the effects due to terrain attributes such as loose soil or slippery surfaces causing the robot to slip, since the actions necessary to stabilise the system might not be defined in the set of behaviours.

Stable footfall generation for a known terrain is computationally expensive due to the high dimensionality of the planning space. When a slip occurs, the planner often has to recompute the footfall from a new post-slip configuration. In this paper, we propose an algorithm to detect and arrest the slip in real-time without knowledge of the local terrain before the body moves to an unstable configuration, potentially damaging itself, or to a configuration from where recovery is difficult.

2 Technical Approach

The complexity of the legged robot locomotion on uneven terrain comes from the high degrees of freedom that need to be controlled using imprecise knowledge of the environmental interaction during the robots gait. The challenge arises not only for planning footfalls on a high DoF robot but also while executing the footfalls in the presence of unexpected events. While unknown terrain poses a

great challenge to legged locomotion, even with full knowledge of the terrain, the robot encounters inherent slips due to loose gravel, slippery vegetation, uneven surfaces etc. Many approaches [9][6][10][4] have attempted to provide stability to the body by taking corrective actions to control the slipping joint or the leg. While individual controls can be added to each leg to prevent its slip, often the stability of the whole body is dependent on all leg positions in a coupled manner. Fixing the slip of one leg may not inherently provide stability. Due to this we need to provide control in the higher dimensional space.

In this preliminary study we take the first steps towards developing a framework for high-dimensional control of the hexapod to prevent slipping in planar terrains. Slipping on a simplified planar terrain model gives us an opportunity to study the high-dimensional slip control without the complexity of unknown terrain.

2.1 Formulation

In this study we focus on a hexapod robot with each leg having 3 actuators giving 18 joints to control. The state space C for the hexapod then lies in 18 dimensions. C consists configurations which satisfy the joint limits, without considering collisions of the vehicle with itself or the environment. Even with tight joint limits, not all configurations in the state space C are valid for stable navigation.

In this work we only look at quasi-static walking gaits, i.e., at any configuration of the robot during gait execution, the robot is inherently stable. While we ignore any dynamic gaits, we believe that our framework lends to an extension for dynamic gaits in a straightforward manner.

A main characteristic of the hexapod leg slip during gait execution is the foot tip position moving away from the planned footfall during the support phase of the gait cycle. When this happens, the support polygon gets skewed and often leads to instability of the whole robot. Another characteristic of leg slip is sudden change in body orientation. This can be detected in body roll, pitch or a combination of both depending on the number and position of the slipping legs. In this study we check for the position of the foot tips and the body orientation compared to a small tolerance around a desired foot tip and body orientation to satisfy the stable configuration. While this check could be extended to the rate of change of the tip position and orientation error, in this study we focus on the measure of error and not its rate of change.

To satisfy the footfall position and the body orientation, we use the 6-dimensional world task space T, in which we define constraints on the tip position and body orientations that limit the planning in C to desired tip and body orientations in T. These constraints generate a desired region T_{constr} inside T. We define constraining cuboids T_i with edges d_x, d_y and d_z for the desired tip position for each leg i, capturing the full span and an operational tolerance of the tip position during the gait cycle. Furthermore, we formally define two frames: a body frame B and a local frame L both with origins at the centre of the robot. The center could be the center of mass or any other suitable origin on the body. L is a 3-dimensional frame with its x, y plane being parallel to the ground plane.

The body frame B is defined such that the x, y plane is congruent to the plane through the body. Note that the tip positions are expressed in L. The orientation of the body is therefore defined as the orientation of B about L.
Let θ_r and θ_p be B's roll and pitch angle about L. By imposing constraints Θ on θ_r and θ_p, we define

$$T_{constr} = (T_n, \Theta) \tag{1}$$

with $T_n = (T_1, ..., T_6)$, where $(T_1, T_2, ...)$ are the space of allowed tip position for the corresponding legs. Figure 2 illustrates the constraint cuboids around the initial tip positions and the body orientation constraints. Any configuration in C that does not satisfy the tip position or body orientation constraints are discarded during planning, thus, T_{constr} induces a manifold M_{constr} such that

$$M_{constr} = \{c \in C \mid f_C^T(c) \in T_{constr}\} \tag{2}$$

with f_C^T being a mapping from C to T. We call a configuration c *stable* if $c \in M_{constr}$, *unstable* otherwise.

During a stable gait, the robot's foot tip positions follow a precomputed trajectory. For a tripod gait, 3 legs move at a time while for a wave gate one leg moves at a time. The stationary legs provide stable support for the body. In this study, we focus on the tripod gait, but the approach is applicable to any other gait with minimal modifications to our implementation.

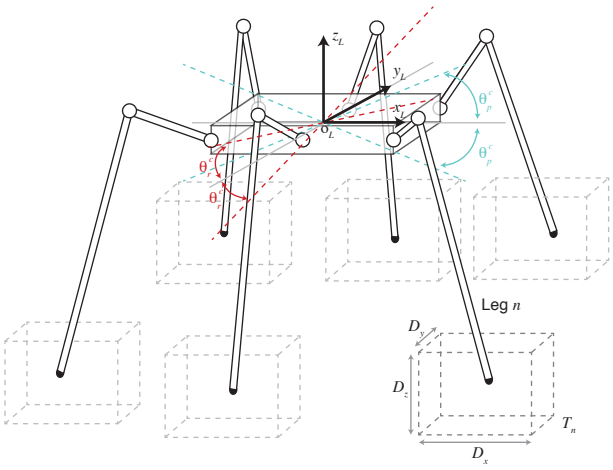


Fig. 2. Foot tip constraint cuboids T_{1-6} $(D_x \times D_y \times D_z)$ and body orientation constraints $\Theta = (\theta_r^c, \theta_p^c)$ shown with respect to configuration of the hexapod robot.

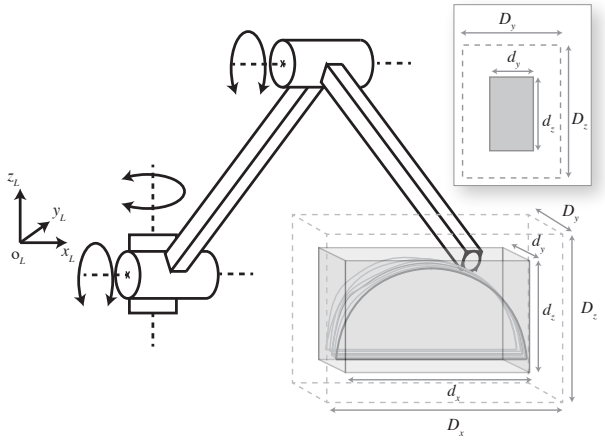


Fig. 3. Detailed illustration of foot tip constraints with cuboid centred at the initial position of foot tip.

2.2 Solution Approach

We start with the assumption that any stable configuration has a neighbourhood in C that is stable. During the stable gait execution, the body moves through a periodic cycle of stable states $c_i \in M_{constr}$.

During a slip event, as the leg tip moves closer to the boundaries of T_{constr}, the robot configuration $c_{current}$ moves closer to the boundary of M_{constr}. Our overall objective of stabilisation is to keep the body configuration inside M_{constr} and react, when the configuration $c_{current}$ travels outside M_{constr}. To do so, we cache known stable configurations in the neighbourhood of the gait cycle in M_{constr} offline and at run time, when the robot slips, quickly find a path from $c_{current}$ back into a stable configuration $c_s \in M_{constr}$. One problem with that approach is that the configuration c_s might be too close to the boundaries of M_{constr} such that from c_s, even a slight disturbance might push the configuration outside M_{constr} again. In order to prevent such situations, we try to restrict c_s to a much smaller region within C which has a sufficient distance to the boundaries of M_{constr}. This is achieved by adding an operational tolerance to T_{constr} yielding $T_{stable} = (T_n^*, \Theta)$. T_n^* consists of a second set of tip constraint cuboids $(T_1^*,\ ...,\ T_6^*)$ with dimensions $d_x,\ d_y,\ d_z$, where $0 < d_x, d_y, d_z < D_x, D_y, D_z$. Figure 3 illustrates the second set of tip constraint cuboids. T_{stable} induces a smaller manifold $M_{stable} \subset M_{constr}$.

Stable configuration caching We are interested in stable configurations which are close to the known stable configuration cycle. Thus, instead of randomly selecting a configuration inside M_{stable} to connect to, the algorithm selects a configuration $c \in M_{stable}$ within a certain neighbourhood of the current configuration cycle. However, sampling M_{stable} can be a difficult task. The shape and

properties of M_{stable} depends on the dimensions of the tip position constraint cuboids defined in T. A naive sampling technique would be to use rejection sampling by uniformly picking a sample $c \in C$ and check whether

$$f_C^T(c) \in T_{stable} \tag{3}$$

This sampling techniques samples the full state space C and rejects samples which lie outside M_{stable}. Since the volume of M_{stable} is unknown *a priori*, it could be the case that it is only a small fraction of the volume of C. In those cases, rejection sampling fails to sample a number configuration candidates inside M_{stable} within a reasonable amount of time. Therefore we generate a point cloud P of configurations inside M_{stable} offline, from which the algorithm can select a configuration to connect to during run-time.

Another approach is to grow a space filling tree, in our case an *RRT*, from a known stable configuration (possibly an initial state or a known home state) inside M_{stable}. The nodes generated by *RRT* are then used as M_{stable} samples. This techniques quickly generates n samples inside M_{stable}, however, it suffers from the same limitations of *RRT*. Despite its probabilistic completeness, *RRT* performs poor in complex environments with narrow passages. Several techniques have been proposed to improve sampling under the occurrence of narrow passages, such as filtering or adaptive sampling techniques [11] [12] [13] or retraction based approaches [14] [15] [16] [17] [18].

We are especially interested in regions inside M_{stable} which can be reached from a known path in M_{stable} defined by a gait cycle of the vehicle, not necessarily a single initial configuration. For this we pick n uniformly spaced samples from the configuration cycle while the robot performs a specific gait (e.g. tripod gait). These n samples are then used as seeds for an *RRT* forest with n trees. From each seed a tree is grown inside M_{stable} with a predetermined number of nodes, using the standard *RRT* algorithm [19]. The nodes of the generated *RRT* forest form a point cloud P within M_{stable}. These points are potential candidates the system can connect to when the task space constraints are violated. Figure 4 illustrates the offline generation of P using *RRT*.

Run time stabilisation During run-time, when the current configuration $c_{current}$ is inside M_{constr}, the algorithm populates a list A with sets of stable configurations located in the neighbourhood of the current configuration $c_{current}$ (figure 5). This is done by sampling k points from P that are within a hypersphere with radius r around $c_{current}$. Recall from 2.2 that the P entirely lies within M_{stable}. The sampled set of points $P^k \subset P$ is then appended to the list. Algorithm 1 shows the population of the list during run-time. The function *samplePointcloud(r)* selects $P^i \subseteq P$ such that

$$P^i = \{p \in P \; s.t \quad ||p - c_{current}|| \leq r\} \tag{4}$$

where $||.||$ is a standard Euclidean distance metric. Out of P^i, *randomSelect(set, k)* randomly selects k points. This set $P^k \subset P^i$ is appended to the list of stable sets.

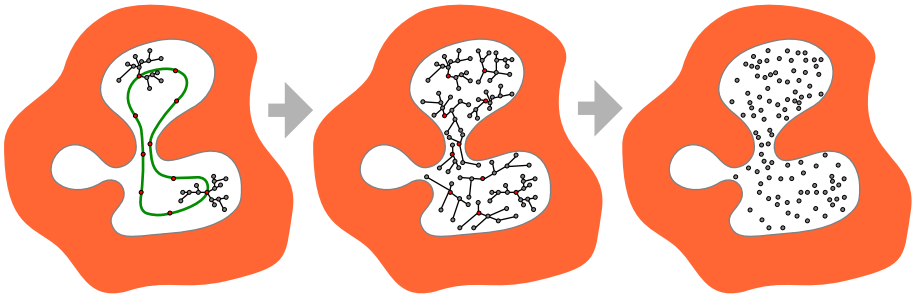


Fig. 4. Illustration of the sampling method based on the *RRT* forest approach to obtain a point cloud of stable configurations. The outer shading shows the area within C which are unstable regarding the task space constraints T_{stable}. The green line shows the path inside M_{stable} for a specific gait cycle. The red dots are the seeds from which the *RRT* trees are grown inside M_{stable}

Algorithm 1 populate_list

1: $A \leftarrow [\,]$
2: **while** $c_{current}$ *is stable* **do**
3: $\quad P^i \leftarrow samplePointcloud(r)$
4: $\quad P^k \leftarrow randomSelect(P^i, k)$
5: $\quad P^k.append(c_{current})$
6: $\quad A.append(P^k)$

Note that the population of the list only happens when $c_{current} \in M_{constr}$. If this is not the case, the *populate_list* algorithm is interrupted until the $c_{current}$ becomes stable again.

This list serves as a source for stable configurations the stabilisation algorithm tries to connect to when the configuration becomes unstable. If this is the case, the algorithm selects the last element A_n of the list (the last set of known stable configurations). Within A_n, the algorithm selects the first configuration as c_{stable} (line 4 in algorithm 2). Since the structure and connectivity of M_{constr} is unknown, instead of joining the states with a direct path, we use *RRTConnect* [20] to find a path from $c_{current}$ to c_{stable} (function *findStablePath* in algorithm 2). For details of *RRTConnect*, the reader is referred to [20] After finding a path p connecting $c_{current}$ and c_{stable} the system executes p (function *execute(stablePath)* in algorithm 2). In some cases, executing p doesn't result in the system to obtain a configuration c such that $c \in M_{stable}$, due to unconsidered interactions of the robot with the ground. Instead c might still be outside M_{constr} after executing p. If this is the case, the algorithm successively selects $c_i \in A_n$ as c_{stable}, calculates p from $c_{current}$ to c_{stable}, and executes p. In case c is still critical after connecting to each $c_i \in A_n$, the next set A_{n-1} in A is chosen and the algorithm tries to connect to the c_i's within that set. This is repeated until the system is in a stable configuration again.

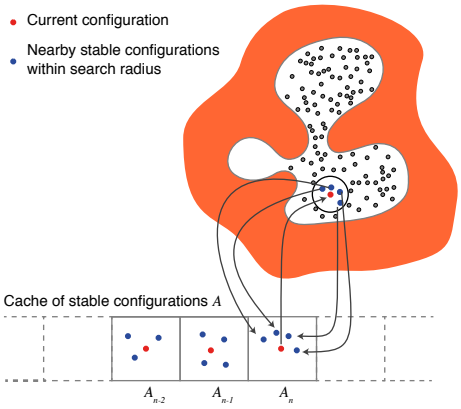


Fig. 5. This figure shows how the cache of stable configurations is populated at runtime with points from the point cloud P. Note that the current configuration (red dot) does not necessarily have to be inside M_{stable}. The caching of stable configurations happens as long as the current configuration is inside M_{constr}

Algorithm 2 connect_to_c_stable

```
1: listIndex ← A.last()
2: setIndex ← 0
3: while c_current is critical do
4:     c_stable ← A[listIndex][setIndex]
5:     stablePath ← findStablePath(c_current, c_stable)
6:     execute(stablePath)
7:     setIndex ← setIndex + 1
8:     if setIndex = k then
9:         listIndex ← listIndex − 1
10:        setIndex ← 0
```

3 Experimental Setup

3.1 Hardware

Our experimental setup consists of a modified PhantomX hexapod kit by Trossen Robotics [1] using Dynamixel AX-18 servosmotors (3 per leg, 18 in total). We have added a Pandaboard embedded computer running Robot Operating System (ROS) and an SBG IG-500N attitude and heading reference system (AHRS) on to our hexapod platform (figure 1(b)). The length of the tibia links have been extended (140 mm → 405 mm) in order to gain a broader range of unstable situations. The robot's body orientation is measured using the IMU of the AHRS. The joint angles of the servomotors are received at a rate of 35 Hz and the IMU data is received at a rate of 100 Hz. Figure 6(a) shows the control and data flow between the hardware components.

[1] http://www.trossenrobotics.com

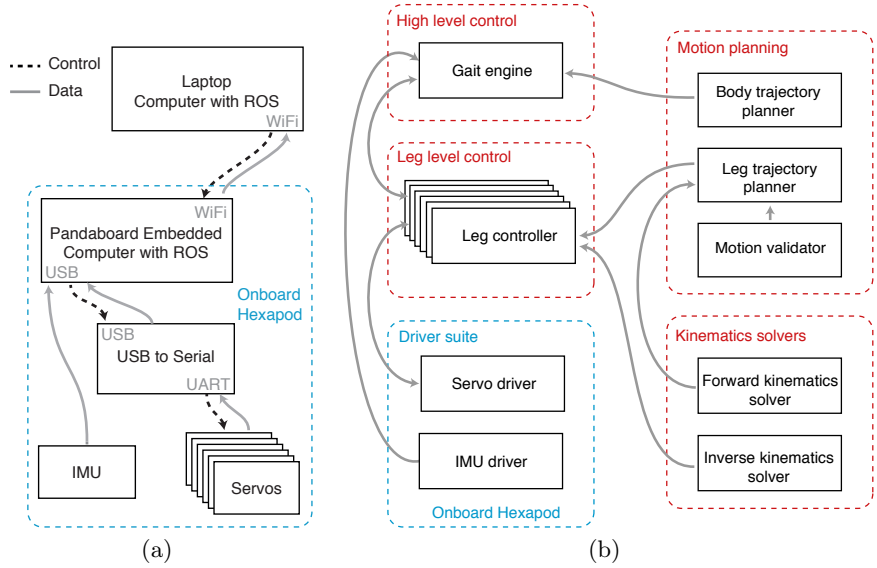


Fig. 6. Hardware (a) and software (b) block diagrams showing control and data flows of the hexapod system.

3.2 Software architecture

The software architecture in which the proposed stabilisation method is embedded is shown in figure 6(b). It consists of a high level gait engine which generates the body locomotion during run-time, by defining desired tip positions for a given gait pattern. These tip position goals are sent to the leg controllers which calculate a trajectory from the current tip position of leg i to the commanded tip position, using a leg trajectory planner and an inverse kinematics solver. For the proposed stabilisation mechanism, the gait engine module monitors the current state of the system (joint angles from the servomotors and body orientation obtained from the on-board IMU) and utilises OMPL (Open Motion Planning Library) to determine if a state is stable or not. It also uses OMPL and its *RRT-Connect* implementation to compute a path in M_{constr} from a configuration outside M_{constr} to a stable one inside M_{stable}.

3.3 Constraint dimensions

The dimensions of the foot tip constraint cuboids have an significant influence on the performance, since these dimensions determine the time it takes for the system to detect a leg slip. If the constraints are too narrow, even slight deviations of the tip positions will lead to a slip detection. On the other hand, if the dimensions of the cuboids are too large, a slip gets detected too late for the system to stabilise the robot. The dimensions of the foot tip constraint cuboids T^*_{1-6} and T_{1-6} used during our experiments are $d_x = 100\,\text{mm}$, $d_y = 60\,\text{mm}$, $d_z = 140\,\text{mm}$

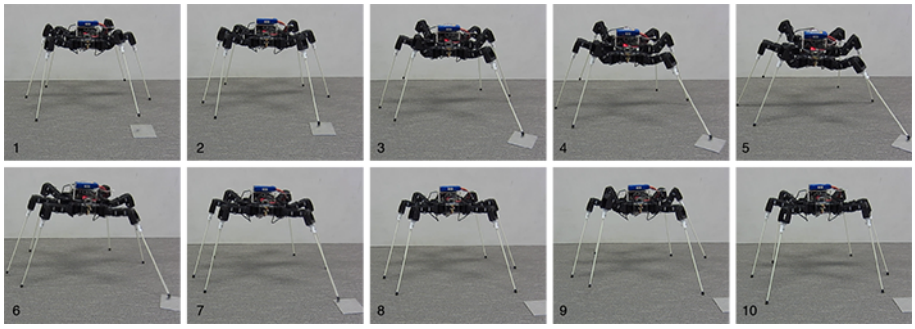


Fig. 7. A sequence of frames showing an experiment where the front left foot of the robot steps on an aluminium plate causing the leg to slip away during a gait (top row). Our stabilisation system allows the robot to recover and continue the gait (bottom row).

and $D_x = 120\,\text{mm}$, $D_y = 80\,\text{mm}$, $D_z = 160\,\text{mm}$ respectively. Body orientation constraints Θ used during our experiments are $\theta_r^c = \pm 5°$and $\theta_p^c = \pm 6°$.

3.4 Performance metrics

In order to be able to define metrics which measure the quality of the proposed stabilisation approach, we need to gain insight about what happens with the robot during a leg slip when no stabilisation is performed. A leg slip induces unexpected movement of the body, resulting in significant variances of the body orientation, as it can be seen in figure 8.

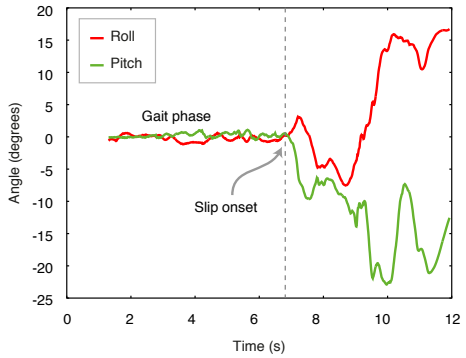


Fig. 8. This figure shows the body's roll (red) and pitch (green) orientation in degree during a gait phase and a leg slipping event without stabilisation

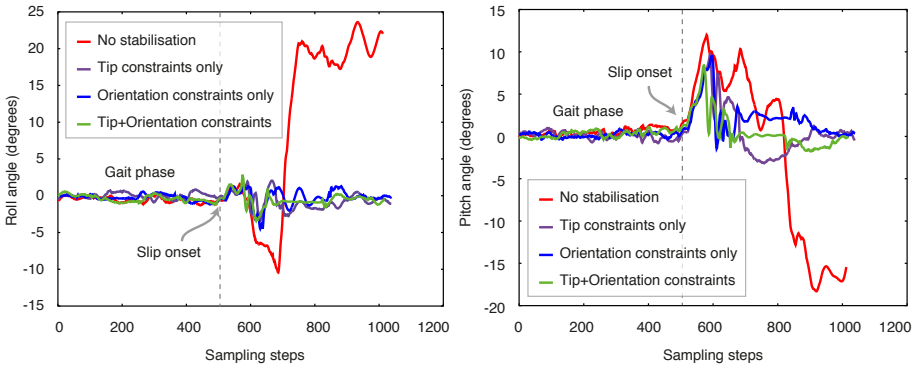
These uncontrolled variances in the body's orientation angles can significantly impact the stability of the robot. Therefore we aspire to limit sudden changes

in the orientation of the body. In other words, the standard deviation of the orientation of the body frame B about the local frame L gives us a metric regarding the stability of the robot.

Another performance metric is the time t_{rec} it takes the system to stabilise the robot after a leg slip has been detected. This is the time interval between a stability criterion being violated to the time the robot is able to continue its locomotion with the specified gait pattern. The longer the robot continues to slip, more difficult it is to recover to a stable state and it becomes more vulnerable to damage either by toppling over or stripping gears. t_{rec} is calculated as

$$t_{rec} = t_c - t_s \tag{5}$$

where t_c is the time when the robot continues its locomotion, and t_s the time when a leg slip gets detected. Note that t_{rec} includes all stabilisation attempts since the system may perform multiple stabilisation attempts when its still in an unstable state after a particular recovery phase.



(a) (b)

Fig. 9. Body roll (a) and pitch (b) during a leg slipping event without stabilisation and with stabilisation using tip, orientation and both constraints.

3.5 Experimental procedure

In our experiments the hexapod walks straight using an alternating tripod gait. During its locomotion we let it step on a slipping obstacle (a small aluminium plate with a low friction coefficient) with a specific leg, causing the interacting leg to slip away from its intended foothold position (figure 7). This causes the whole robot to become unstable and, in certain cases, without our stabilisation system, the body to topple over. We repeat that experiment three times for each leg. This experimental design is repeated for each of the two task-space

constraints (tip position cuboids and orientation constraints) and a combination of the two constraints. We also conducted a set of experiments where the stabilisation system was turned off. The standard deviations of body roll and pitch angles are recorded for each of the experimental runs and presented in the following section along with a comparison of recovery times for different stability constraint combinations.

4 Results and Insights

Tables 1, 2, 3 and 4 show the standard deviation of the body pitch and roll angles with respect to the local frame L during a leg slip and the subsequent stabilisation phase. The first column refers to the leg which was slipping. The second and third column refers to the standard deviation of the respective angular direction.

The results in table 1 were obtained without any stabilisation, whereas for the tables 2, 3 and 4 the proposed stabilisation approach used different task space constraints (orientation constraints only, tip position constraints only, orientation and tip position constraints).

It can be seen that the standard deviation for the roll and pitch angles are significantly reduced using the proposed stabilisation mechanism. Using the orientation constraints only had a higher overall standard deviation in roll and pitch angles, followed by the tip position constraints. The best results were obtained by using both types of task space constraints.

Table 1. Standard deviation of body roll and pitch (θ_r and θ_p) with no stabilisation for a series a leg slipping experiments.

Slipping leg	σ_{θ_r}	σ_{θ_p}
Front left	3.65 °	4.02 °
Front right	4.75 °	5.21 °
Middle left	3.45 °	4.12 °
Middle right	4.89 °	4.76 °
Rear left	7.58 °	8.93 °
Rear right	5.51 °	6.96 °

Table 2. Standard deviation of body roll and pitch (θ_r and θ_p) with stabilisation using orientation constraints only for a series a leg slipping experiments.

Slipping leg	σ_{θ_r}	σ_{θ_p}
Front left	1.03 °	3.19 °
Front right	1.58 °	2.36 °
Middle left	1.55 °	3.09 °
Middle right	1.00 °	2.65 °
Rear left	2.26 °	2.26 °
Rear right	3.01 °	2.80 °

Table 3. Standard deviation of body roll and pitch (θ_r and θ_p) with stabilisation using tip constraints only for a series a leg slipping experiments.

Slipping leg	σ_{θ_r}	σ_{θ_p}
Front left	1.76 °	3.17 °
Front right	1.30 °	2.20 °
Middle left	1.92 °	1.59 °
Middle right	1.70 °	2.65 °
Rear left	1.28 °	2.08 °
Rear right	1.75 °	1.38 °

Table 4. Standard deviation of body roll and pitch (θ_r and θ_p) with stabilisation using tip constraints and orientation constraints for a series a leg slipping experiments.

Slipping leg	σ_{θ_r}	σ_{θ_p}
Front left	1.74 °	2.73 °
Front right	1.02 °	1.65 °
Middle left	1.27 °	2.07 °
Middle right	1.57 °	1.82 °
Rear left	1.72 °	2.63 °
Rear right	1.37 °	1.48 °

Table 5. Mean recovery times t_{rec} for leg slip recovery for each leg with different constraints along with the mean number of stabilisation attempts given within brackets.

Slipping leg	t_{rec} for tip and orientation constraints	t_{rec} for tip constraints only	t_{rec} for orientation constraints only
Front left	0.85 s (1.0)	1.06 s (1.3)	1.53 s (1.6)
Front right	0.89 s (1.0)	1.16 s (1.0)	1.93 s (1.6)
Middle left	1.26 s (1.0)	0.96 s (1.3)	1.92 s (1.6)
Middle right	0.52 s (1.6)	1.66 s (2.6)	1.79 s (2.0)
Rear left	0.72 s (1.6)	4.34 s (3.6)	1.87 s (2.0)
Rear right	1.49 s (1.3)	2.99 s (3.0)	1.96 s (1.6)

Table 5 shows the time t_{rec} it takes for the system to get back to a stable state after a leg slip occurred and the number of stabilisation attempts, using different constraints. This includes the time $t_{connect}$ it takes for *RRTConnect* to find a path from a state that violates the task space constraints to a state inside the stable mainfold M_{stable}, and the time t_{exec} the system takes to execute that path. On average, *RRTConnect* took 0.11 s until the algorithm found a path back to M_{stable}. The table also gives the mean number of stabilisation attempts after a leg slip for each leg. The recovery time using only the tip constraint cuboids was much higher compared to using orientation constraints alone when a rear leg was slipping. We believe that this was due to the direction of motion of the rear foot tips during the support phase in the gait cycle being in the

same direction as the slip. This makes the overall slip much severe compared to the front or middle legs. Therefore, slips of the rear legs took a greater number of stabilisation attempts before the robot was able to continue its locomotion resulting in an overall longer recovery time for slips of the rear legs when using the tip constraint cuboids only. Using orientation constraints appeared detect and arrest the slip event much quicker in such instances. The stabilisation system showed best overall performance when both the tip constraint cuboids and the orientation constraints were used as seen from the results.

5 Conclusions

We presented a framework for real-time stabilisation of a high dimensional multi-legged robot. An experimental evaluation of this framework was performed using a hexapod robot and the results demonstrated the method effectively detects and recovers from unexpected events such as leg slip. The standard deviation of roll and pitch of the robot's body was used as a metric for stability. The stability of the robot improved by 2× when the proposed method was used, with a reduction of 3.27°→ 1.45° standard deviation for roll angle and a reduction of 3.94°→ 2.06° standard deviation for pitch angle. We also presented results for reaction time of the system when using different stability constraints based on foot tip positions as well as the body roll and pitch angles and demonstrated that the best results are achieved when both types of constrains are used for detecting instability at run-time. With the proposed real-time stabilisation system, the hexapod robot was capable of successfully recovering from unexpected leg-slip events and re-commence locomotion without explicit knowledge of the local terrain. Our current study is limited to planar terrains of zero elevation. We believe that our approach can easily extend to elevated planar slopes with the help of an onboard imu. We are currently extending our approach to work on elevated planar surfaces. We plan to incorporate system dynamics with a better model of the hexapod platform and knowledge of the terrain from an onboard perception module to make our approach robust to challenging environments in the near future.

References

1. Lee, T.T., Liao, C.M., Chen, T.: On the stability properties of hexapod tripod gait. IEEE Journal of Robotics and Automation **4**(4) (1988) 427–434
2. Hauser, K.K., Bretl, T., Latombe, J.C., Harada, K., Wilcox, B.: Motion planning for legged robots on varied terrain. International Journal of Robotics Research **27**(11-12) (2008) 1325–1349
3. Bretl, T., Rock, S.M., Latombe, J.C.: Motion planning for a three-limbed climbing robot in vertical natural terrain. In: proceedings of the IEEE International Conference on Robotics and Automation (ICRA). (2003) 2946–2953
4. Belter, Dominik; Skrzypczynski, P.: Integrated motion planning for a hexapod robot walking on rough terrain. In: proceedings of the IFAC World Congress. Volume 18. (2011) 6918–6923

5. Belter, D., Skrzypczynski, P.: Posture optimization strategy for a statically stable robot traversing rough terrain. In: proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). (2012) 2204–2209
6. Kerscher, T., Rönnau, A., Ziegenmeyer, M., Gassmann, B., Zoellner, J., Dillmann, R.: Behaviour-based control of a six-legged walking machine LAURON IVc. proceedings of the 11th International Conference on Climbing and Walking Robots (CLAWAR) (2008) 8–10
7. Rönnau, A., Kerscher, T., Ziegenmeyer, M., Zollner, J., Dillmann, R.: Adaptation of a six-legged walking robot to its local environment. In Kozowski, K., ed.: Robot Motion and Control 2009. Volume 396 of Lecture Notes in Control and Information Sciences. Springer London (2009) 155–164
8. Best, G., Moghadam, P., Kottege, N., Kleeman, L.: Terrain classification using a hexapod robot. In: proceedings of the Australasian Conference on Robotics and Automation (ACRA). (2013)
9. Wettergreen, D., Thorpe, C.: Developing planning and reactive control for a hexapod robot. In: proceedings of the IEEE International Conference on Robotics and Automation (ICRA). (1996) 2718–2723
10. Lewinger, W.A., Quinn, R.D.: A hexapod walks over irregular terrain using a controller adapted from an insect's nervous system. In: proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). (2010) 3386–3391
11. Boor, V., Overmars, M.H., van der Stappen, A.F.: The Gaussian sampling strategy for probabilistic roadmap planners. In: proceeding of the IEEE International Conference on Robotics and Automation (ICRA). (1999) 1018–1023
12. Simon, T., Laumond, J.P., Nissoux, C.: Visibility-based probabilistic roadmaps for motion planning. Advanced Robotics **14**(6) (2000) 477–493
13. Sun, Z., Hsu, D., Jiang, T., Kurniawati, H., Reif, J.H.: Narrow passage sampling for probabilistic roadmap planning. IEEE Transactions on Robotics **21**(6) (2005) 1105–1115
14. Lee, J., Kwon, O., Zhang, L., Yoon, S.E.: SR-RRT: Selective retraction-based RRT planner. In: proceeding of the IEEE International Conference on Robotics and Automation (ICRA). (2012) 2543–2550
15. Zhang, L., Manocha, D.: An efficient retraction-based RRT planner. In: proceeding of the IEEE International Conference on Robotics and Automation (ICRA). (2008) 3743–3750
16. Rodrguez, S., Tang, X., Lien, J.M., Amato, N.M.: An obstacle-based rapidly-exploring random tree. In: proceeding of the IEEE International Conference on Robotics and Automation (ICRA). (2006) 895–900
17. Hsu, D., Snchez-Ante, G., Cheng, H.L., Latombe, J.C.: Multi-level free-space dilation for sampling narrow passages in PRM planning. In: proceeding of the IEEE International Conference on Robotics and Automation (ICRA). (2006) 1255–1260
18. Saha, M., Latombe, J.C.: Finding narrow passages with probabilistic roadmaps: the small step retraction method. In: proceedings of the IEEE/RSJ International COnference on Intelligent Robots and Systems (IROS). (2005) 622–627
19. Lavalle, S.M., Kuffner, J.J., Jr.: Rapidly-exploring random trees: Progress and prospects. In: Algorithmic and Computational Robotics: New Directions. (2000) 293–308
20. Kuffner Jr., J.J., Lavalle, S.M.: RRT-Connect: An efficient approach to single-query path planning. In: proceeding of the IEEE International Conference on Robotics and Automation (ICRA). (2000) 995–1001