Control and Omni-directional Locomotion of a Crawling Quadruped

Douwe Dresscher, Michiel van der Coelen, Jan Broenink, and Stefano Stramigioli

Robotics and Mechatronics, Faculty EEMCS,
University of Twente, 7500 AE Enschede, Netherlands
D.Dresscher@ieee.org, J.F.Broenink@utwente.nl

Abstract. Traversing unstructured environments, (statically stable) legged robots could be applied effectively but, they face two main problems: the *high complexity* of the system and the *low speed of locomotion.* To address the complexity of the controller, we apply a control layer that abstracts the legged robot to an omni-directional moving mass. In this control scheme, we apply the gait generator as proposed by Estremera and de Santos. We present theory to determine the theoretically maximum achievable velocity of a quadruped and compare the (omni-directional) maximum velocity of the selected gait generator with this optimum to validate its performance. For our use case the theoretically maximum achievable velocity is 1 ms^{-1}; in simulations we achieve a velocity for straight movement of maximum 0.75 ms^{-1}. Normal turns with a radius larger than 0.45 m are possible at a velocity of at least 0.1 ms^{-1}; the performance of crab turns is too unpredictable to be useful. The gait generator as proposed by Estremera and de Santos is partially capable of supporting omni-directional movement at satisfactory velocities.

1 Introduction

Traversing unstructured environments, (statically stable) legged robots can be superior to their wheeled and tracked counterparts. However, so far only few have made it to practical applications.

Two main problems that have prevented statically stable legged locomotion from being applied effectively are: the *high* complexity of the system and the *low* speed of locomotion [3,6].

To limit the mechanical complexity of the system, we assume a quadrupedal robot: four is the minimum amount of legs required for statically stable locomotion [4]. To address the complexity of the controller, we present a control scheme where we apply separation of concerns to reduce the complexity. Our approach is a port-based approach which provides a control layer that abstracts the legged robot to an omni-directional moving mass (an admittance) as shown in Fig. 1. By applying a force to the abstracted robot, the resulting velocity of the robot can be controlled (for instance with an impedance controller [1,7]) as shown in

D. Brugali et al. (Eds.): SIMPAR 2014, LNAI 8810, pp. 486–497, 2014.

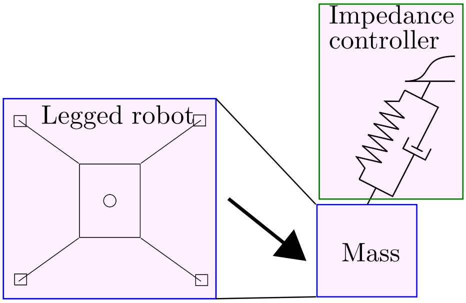


Fig. 1. Abstraction of a legged robot to an omni-directional moving mass that can then, for instance, be controlled by an impedance controller

Fig. 1. This scheme requires a gait controller that ensures that the legs move to support the motion of the robot.

The gait generator as proposed by Estremera and de Santos [2] is capable of generating a gait based on the omni-directional velocity of the robot body and is, for this reason, particularly suitable to be used in the proposed control scheme.

In addition, we present theory to determine the theoretically maximum achievable velocity of a quadruped and compare the speed performance of the selected gait generator with this optimum to validate its performance.

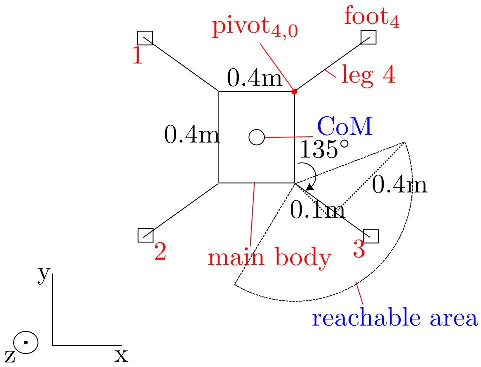
This paper is structured as follows: in Sec. 2, the use case for supporting the theory and evaluating the performance of the gait generator is described. In Sec. 3, the proposed controller structure is presented and explained, including a summary of the gait generator. In Sec. 4, we present theory on the maximum velocity of a quadruped and treat how this applies to the use case. In Sec. 5, the speed performance of the simulated gait generator is presented. In Sec. 6, the simulation results are compared with the theoretical optima and in Sec. 7, conclusions are drawn.

2 Use Case

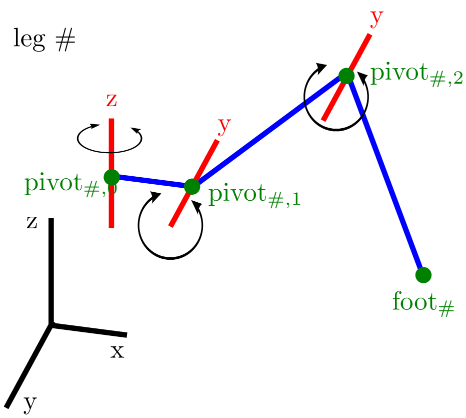
In this work, we employ a use case to clarify theory and evaluate the speed performance of the gait generator. The use case is a quadrupedal robot of which a top view is shown in Fig. 2a and the legs have a configuration as shown in Fig. 2b. The workspace of a leg in the x-y plane is called the *reachable area* (also shown in Fig. 2a).

Throughout this work, we assume that the mass of the legs is negligible compared to the mass of the base. To account for possibly destabilising effects caused by unmodelled behaviour, a safety margin is used. This safety margin is shown as a circle round the Center of Mass (CoM) in Fig. 2a.

For simplicity reasons, a rectangular motion profile is assumed for a step from a starting foothold to a target foothold, P, as shown in Fig. 3. The ground



(a) A top-down perspective of the robot-model. pivot$_{4,0}$ is where leg 4 with a setup shown in Fig. 2b is connected to the main body. CoM refers to the center of mass of the robot.



(b) The leg-setup of the provided robot-model. There are three degrees of freedom. n denotes the leg-number [1..4].

Fig. 2. The robot model

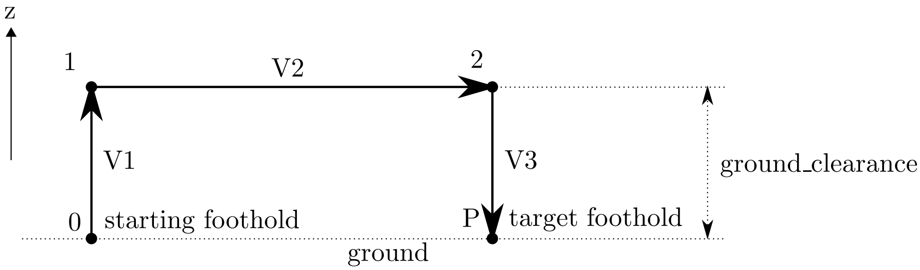


Fig. 3. A step profile. A foot is lifted from point 0 to point 1 with speed V1. The speed between point 1 and 2 is V2. The landing from point 2 to the final foothold P is done at speed V3.

clearance is assumed to be 0.1 m and the velocity at which a leg moves (V1=V2= V3 in Fig. 3) 5 ms^{-1}.

3 Controller Structure: Separation of Concerns

As stated in Sec. 1, we use our controller structure to create a layer that abstracts the legged robot to an omni-directional moving mass. To achieve this several facilities are required as is also shown in Fig. 4:

1. The force that has to be exerted on the robot body needs to be translated to forces that are to be exerted by the legs. (Generalised inverse Jacobian relations)

2. A controller to control the feet over a step trajectory when required (Feet controllers)
3. The step trajectory needs to be generated (Feet trajectory generators)
4. The legs need to be moved such that the movement of the robot body is supported. (Gait generator)
5. Forces to be exerted by the legs need to be translated to forces to be exterted in the joints (Jacobian relations)

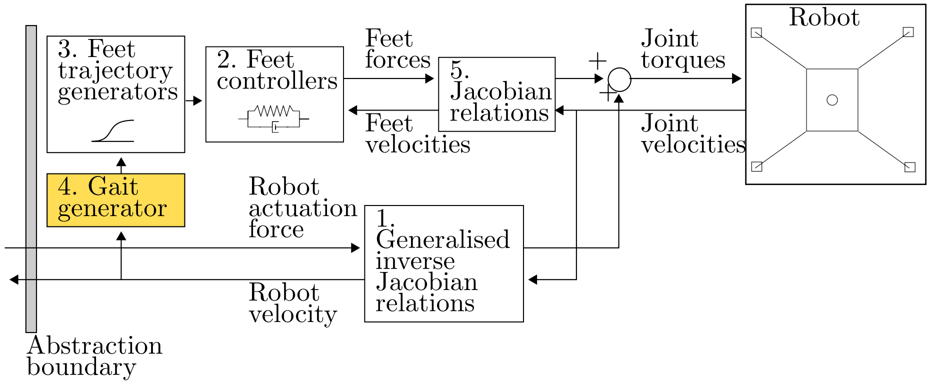


Fig. 4. The abstraction layer with its components

In this work, we assume the feet controllers and feet trajectory generators to be straightforward, and thus we focus on the gait generator.

3.1 Gait Generator

In this Section, we give a short summary of the gait generator.

Two Basic Notions. Two basic notions are used in the summary of the gait generator: the "Kinematic Margin" and the "Transfer Distance".

The Kinematic margin (KM) refers to the distance the CoM can travel in its forward direction until a specific leg is at its physical limit [5]. It is a scalar value that can be visually represented by a line with a length of KM in the direction opposite to the CoM movement and starting at the foot as is shown in Fig. 5. During movement of the CoM, the foot of a leg is assumed to stay in the same location while the rest of the leg moves with the CoM.

The kinematic margin is dependent on the reachable area that is defined by the limits of the leg. The reachable area for the robot model as described in Sec. 2 is shown in Fig. 5. The CoM moves in the direction of the CoM velocity, which causes the reachable area to translate in the same way. After KM meters, the foot will transition out of the reachable area. A minimum value for KM, KM_{min}, is used to denote the smallest kinematic margin among all legs.

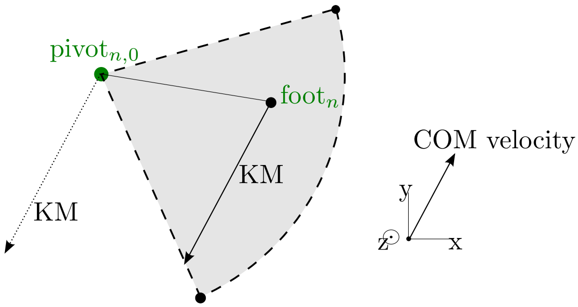


Fig. 5. Visualization of the Kinematic margin (KM)

The Transfer Distance, TD, is the distance that the CoM will travel during the transfer of the foot to P [2]. The minimum value for TD, TD_{min}, is the distance that the CoM will travel when the foot is only moved up and down (also see Fig. 3).

State Machine. The gait generator is built around a state machine as shown in Fig. 6 where, during normal operation, three states (S0, wait; S1, Calculate and; S2, Transfer) are executed sequentially based on three conditions (T01, The next leg can be lifted; T12, Foothold selection successful and; T20, leg transfer complete).

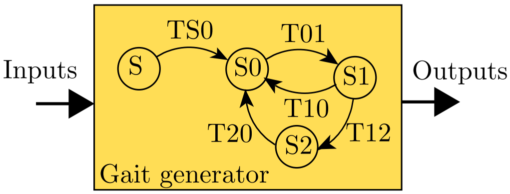


Fig. 6. The main gait-generator state-machine

The output of the gait generator is a leg index for a leg that should be moved and the position where the foot should be placed, the inputs are:

- the location of all feet.
- the location and velocity of the center of mass projection on the xy-plane.

– an indication that the current transfer of a leg has finished.

The three states - S0, S1 and S2 - will now be discussed in detail.

State 0: Wait This is the starting state for the state-machine. All feet are on the ground. A check is made for the ability to lift a leg with stability. A leg is selected based on an order of leg preference:

$$\text{Order of leg preference} = \left[\text{Priority leg},\ \text{lowest KM},\ \ldots,\ \text{highest KM}\right]$$

Here, the first entry is a preferred leg to be transferred that has been determined in the previous state 1. After this, all legs are prioritised based on their kinematic margin: the leg that will reach its kinematic limit sooner has a higher priority.

When the leg preference is determined, the highest priority leg is tested for its ability to be lifted based on the following rule: When the leg is lifted, the center of mass needs to be supported by the other three legs.

When the velocity and forward direction of the vehicle are constant, it is possible to generate a wave gait. A wave gait has superior stability properties [4] and is therefore desirable. In the case of a constant velocity and forward direction of the vehicle, only the highest priority leg is considered for lifting to enforce a wave gait. If the velocity *has* changes, the other legs are considered for lifting in sequence of priority.

The selected leg is named "LT" (Leg to be Transferred). When LT can be lifted with stability - T01 - the state machine transitions to state 1.

State 1: Calculate In this state the next position for the leg to be transferred is calculated. To find a suitable foothold position, several areas are iteratively combined. These areas are named O, D, A, B and C and they are combined (Fig. 7) as follows: First, area O and D are combined.

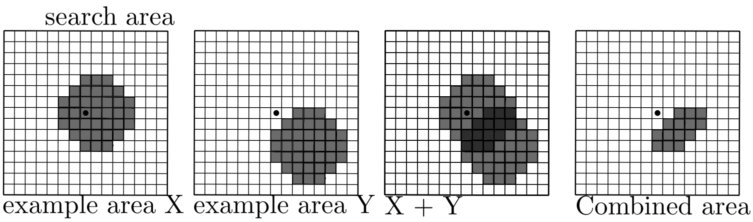


Fig. 7. An example of combining two limiting areas for the footholds named X and Y. The resulting Combined area on the far right shows the collection of all footholds that appear in X as well as Y.

Area O makes sure that placing the leg LT at P does not restrict the next step to be smaller than the current. Area O also takes care that the foothold P is in the workspace of the leg. It can be interpreted as: After placing the foot of LT at P, it should have a KM which is at least TP higher than KM_{min}. Area D can be interpreted as: the footholds that can be reached while the CoM is still in the current support pattern. This condition has to be met in order for the robot to remain stable during the transition of the leg.

If this results in valid footholds, an NLT (Next Leg to be Transferred) is selected based on the following priority:

$$\text{NLT preference} = \left[\text{Next leg in wave gait},\ \text{lowest KM},\ \ldots,\ \text{highest KM}\right]$$

and areas A, B and C are applied. Area A is used to evaluate if leg NLT can be moved if leg LT is placed at position P. Area A can be interpreted as: The point where LT can be placed such that the CoM can travel at least TD_{min} in distance before it either exits the support patters or leg LT reaches its kinematic limit. Area B can be interpreted as follows: it consists of the points P, where can LT can be placed such that the next leg to be transferred, NLT, can be lifted before any of the legs reach their kinematic limit. When a wave-gait leg order is used, knowledge is available about the next three (and further) legs that will be lifted. Area C consist of the points where LT can be placed such that there is enough space for three following feet to be moved.

If this does *not* result in valid footholds, the next leg in the priority sequence is selected as NLT and the step is repeated. If it *does* result in valid footholds, the algorithm can proceed.

Often, more than one suitable foothold is available after combining the areas. To select a foothold from this set of suitable footholds, a criterion can be used. For this work, we use a simple criterion namely: The maximum kinematic margin of the leg (Fig. 8).

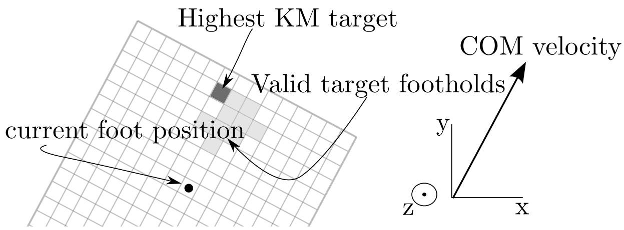


Fig. 8. Foothold selection in the gait generator. The foothold with the highest KM is selected.

If state 1 successfully selects a suitable foothold - T12 -, the state machine goes to state 2. In the case that state 1 fails to find a suitable foothold - T10 - , the state machine returns to state 0.

State 2: Transfer In this state the leg is moved to the target position and the state machine wait until the transition is completed. When the transition is completed, the state machine goes to state 0.

4 Locomotion Speed

In this section, we treat a theory on the theoretically maximum achievable locomotion speed of a crawling quadruped (straight movement only). For this theory, we assume that a crawl gait is used (LF-RR-RF-LR, [4]).

Ideally, a quadruped is designed such that the rear leg that is maximally moved forward can reach the same position as the corresponding front leg that is maximally moved backward. When this is the case, we get consecutive support patterns as shown in Fig. 9. In this image, l is the maximum step size of a leg.

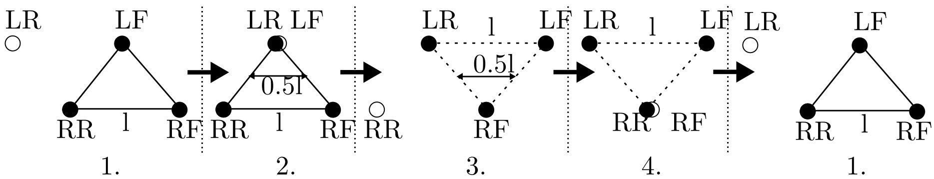


Fig. 9. Consecutive support patterns for a crawl gait when a rear leg, maximally moved forward, can reach a front leg, maximally moved backward. The black filled circles indicate support legs and the white filled circles indicate the leg that is going to make the next step. The black triangles indicate the current support pattern.

Assuming that the CoM travels through the center of the support patterns, in a straight line, two legs have to be moved while the CoM travels $0.5l$ meters. This results in a maximum velocity of the CoM of:

$$V_{max} = \frac{l}{4 * t_{step}} \tag{1}$$

Where V_{max} is the maximum velocity of the CoM, l is the maximum step size and t_{step} is the time it takes to make a step of length l.

It is important to realise that the body of the robot moves when a step is made. For this reason, the actual size of a step is equal to the distance that the body travelled plus the distance that the foot travelled with respect to the body:

$$l = l_{step} + t_{step} * V_{max} \tag{2}$$

such that, the actual maximum CoM velocity becomes:

$$V_{max} = \frac{l_{step}}{3 * t_{step}} \tag{3}$$

In reality, a rear leg that is maximally moved forward often can *not* reach the same position as the corresponding front leg that is maximally moved backward. This is also the case for our use case. This results in consecutive support patterns as shown in Fig. 10. In this image, l is the maximum step size of a leg, d is the

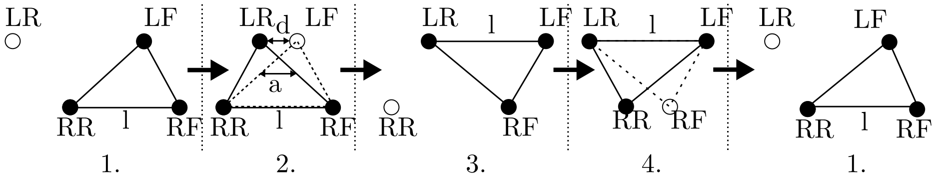


Fig. 10. Consecutive support patterns for a crawl gait when a rear leg, maximally moved forward, can not reach a front leg, maximally moved backward, by a distance d. The black filled circles indicate support legs and the white filled circles indicate the leg that is going to make the next step. The black triangles indicate the current support pattern.

distance between a rear leg's foremost position and a front leg's rearmost position and a is the distance that the CoM can travel while two legs are moved.

The distance a is shorter than $0.5 * l$, namely: $a = 0.5 * (l - d)$. This results in a maximum velocity of:

$$V_{max} = \frac{l - a}{4 * t_{step}} = \frac{l_{step} - a}{3 * t_{step}} \tag{4}$$

For our use case we have:

- $l_{step} = 0.51\ m$
- $t_{step} = 0.14\ s$
- $a = 0.08\ m$

such that we can theoretically achieve a locomotion velocity of:

$$V_{max} = \frac{l_{step} - a}{3 * t_{step}} = \frac{0.51 - 0.08}{3 * 0.14} = 1\ ms^{-1} \tag{5}$$

5 Simulation Results

In this section, we present the simulation results of the locomotion velocity with the simulated gait generator. To simulate the gait generator, a simplified robot model and controller structure are assumed. First of all, we assume that the robot is moving with a fixed velocity and that the "feet trajectory generators" and "Feet controllers" control the feet to move from their starting position to a desired foothold P in $t_{step} = (l_{step} + 0.2)/5$ (see also Sec. 2) where l_{step} is the size of the step. With these assumptions, the controlled system reduces to Fig. 11. To test the maximum velocity of the gait generator for omni-directional movement, simulation runs of 25 s were done to verify stability of the gait for:

1. Straight movement with various crab angles (Fig. 12a).
2. Turning movement, normal and crab-like with various turning radii (Fig. 12b and Fig. 12c).

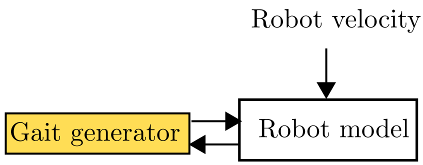
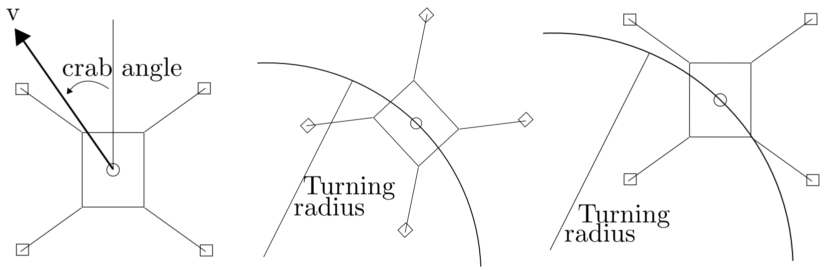
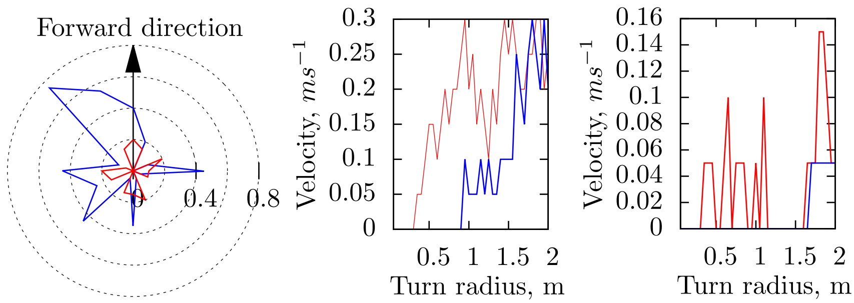


Fig. 11. An overview of the simplified system



(a) Straight motion (b) Normal curve (c) Crab-like curve

Fig. 12. Motion types that are used to test the maximum velocity of the gait generator for omni-directional movement



(a) Straight motion under various crab angles (b) Normal turn with various radii. (c) Crab turn with various radii.

Fig. 13. Simulation results with wave gait (blue) and with random gait (red)

The simulation results are shown in Fig. 13.

Since the simulation does not include any dynamic effects, no safety margin was used on the CoM location (Sec. 2). Doing so is expected to give the best performance: it is expected that including a safety margin causes a more conservative, and thus slower, gait to be generated. Furthermore, the starting position of the CoM in the support patters was moved 0.1 m to the rear as shown in Fig. 14. This is expected to cause the direction-dependent performance in the velocity of the gait.

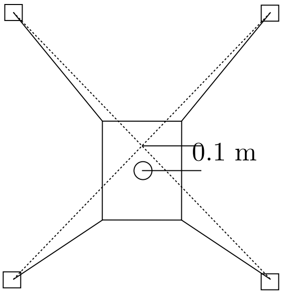


Fig. 14. In the simulations, the CoM was placed 0.1m to the rear with respect to the support patterns (initial condition)

6 Evaluation

In theory, we showed that we could achieve a velocity of 1 ms^{-1} for straight motion in the forward direction, with a wave gait. In simulations we have achieved velocities up to 0.75 ms^{-1} (Fig. 13a) with a wave gait, but not in the forward direction. In the forward direction we achieved 0.4 ms^{-1} (Fig. 13a). These results are in the same range as the results of Estremera and de Santos [2] for straight motion and 40% and 75%, respectively of the theoretically maximum achievable velocity.

It is shown in the simulations that, for straight motion, the overall velocity with a wave gait is significantly better than with a random gait, the latter showing a poor performance.

For strait motion we see a dependency on the crab angle, as expected. The high maximum velocity when moving at a crab angle of $\pi/4$ is curious. We noticed that the initial conditions have a significant effect on the ability of the gait generator to start a gait; we expect that this high velocity is due to a "lucky coincidence" of initial conditions.

The simulations show stable results for making a normal turn at a velocity of at least 0.1 ms^{-1} at a radius of more than 0.45 m (Fig. 13b). The performance for crab turns is too unpredictable to be useful (Fig. 13c). For turns, the performance of the wave gait is significantly worse than the random gait.

7 Conclusions

In this work, we have presented a controller structure in which we applied separation of concerns to address the complexity. In this controller we apply the gait generator of Estremera and de Santos [2]. We have presented theory on the maximum achievable velocity with a statically stable crawl gait and tested the maximum velocity of the gait generator for omni-directional movement resulting in 75% of the theoretical maximum.

The performance of the gait generator is strongly dependent on the crab angle. The highest performance is achieved at a crab angle of $\pi/4$.

Normal turns at low velocities are possible but crab turns show poor performance.

The gait generator is partially capable of supporting omni-directional movement at satisfying velocities.

We want to apply the results of this paper in the control strategy by restricting the motion of the robot to velocities where the gait generator has a good performance. Furthermore, future work includes synthesis of the other components in the control strategy and experiments on a quadruped setup.

References

1. Arevalo, J.C., Garcia, E.: Impedance control for legged robots: An insight into the concepts involved (2012)
2. Estremera, J., de Santos, P.G.: Generating continuous free crab gaits for quadruped robots on irregular terrain. IEEE Transactions on Robotics 21(6), 1067–1076 (2005)
3. Hardarson, F.: Locomotion for difficult terrain. Tech. rep., Royal Institute of Technology, Stockholm (1997)
4. McGhee, R.B., Frank, A.A.: On the stability properties of quadruped creeping gaits. Mathematical Biosciences 3, 331–351 (1968)
5. McGhee, R.B., Iswandhi, G.I.: Adaptive locomotion of a multilegged robot over rough terrain. IEEE Transactions on Systems, Man and Cybernetics 9(4), 176–182 (1979)
6. Seeni, A., Schafer, B., Rebe, B.: Robot Mobility Concepts for Extraterrestrial Surface Exploration. Tech. rep., Institue of Robotics and Mechatronics, International space university, Wesseling, Illkirch Graffenstaden (2008)
7. Yoneda, K., Iiyama, H., Hirose, S.: Sky-hook suspension control of a quadruped walking vehicle. In: Proceedings of the 1994 IEEE International Conference on Robotics and Automation, pp. 999–1004. IEEE (1994)