Learning Robot Locomotion

Achim Stein
Department of Computer Science
Technical University of Darmstadt
steinachim@gmx.de

Abstract

With the growth of computational power in recent years, complex autonomous robot systems have become a possibility. To increase their flexibility and adaptability, learning mechanisms are necessary for robot control. In this paper, I present two recent legged robot-systems that use machine-learning in their foothold-selection process. It was found that currently both systems have a number of issues, including high failure rates in autonomous navigation and the necessity of exact maps and external positioning information. The combination of both systems may lead to an overall improvement in performance.

1 Introduction

Over the last decade, computing power has grown exponentially, while the computers themselves have become smaller and smaller in size. This trend has allowed autonomous mobile robots to become much more powerful in their sensing and computation abilities and paves the way for a future in which robots as everyday-helpers have become a common occurrence. In order to make this possible, our current robot setups must become more adaptable and able to learn from their environment to cope with changing or completely unknown situations.

1.1 Different ways of controlling robots

In the past, most robot systems used *adaptive control*, which, according to [7], focuses basically on convergence without failure. This concept is based on the idea that during the learning phase, no truly incorrect or impossible decisions can be made. The adaptive control approach is relevant especially in scenarios where failures may lead to damaging the robot (e.g. learning to walk without the help of simulations). Only more recently, the focus has shifted towards *learning control*. This control method is oriented more along the lines of human learning by trial and error. The advantage here is the possibility to have more general goals for the optimization process than with adaptive control. Of course, the "error" part needs to be contained as to keep the robot within its safety margins.

1.2 Different classifications for robot learning

In general, there are three main axis that can be used to define a robot learning system. They are shown in Figure 1.

The first axis describes the difference between *direct* and *indirect* control, which is the difference in what is to be learned. In a model-based (or indirect) learning scenario, the robot learns a model describing its own design and abilities which is used for control afterwards. Model-free / direct approaches, on the other hand, do not have the model as an intermediate layer. Instead, the control parameters are learned directly.

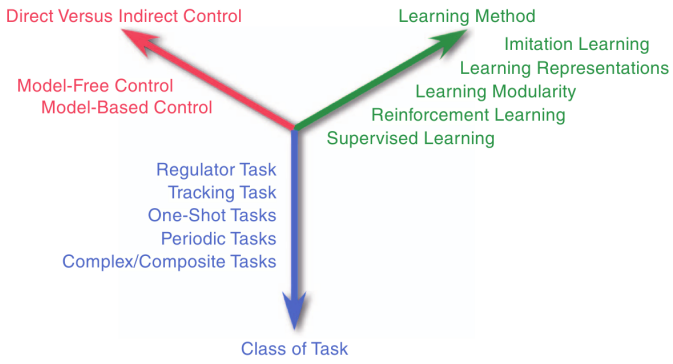

Figure 1: The three dimensions of learning classifications (see [7])

The second axis defines the learning method used. Examples of these are *reinforcement learning* and *imitation learning*, both of which will be used later in this paper.

Finally, shown along the third axis, the task to be learned has a heavy impact on the learning system used. In the course of this paper, we look at complex tasks as the selection of correct footholds is a complex endeavor that is made up of different smaller tasks.

1.3 Overview over this paper

The aim of this paper is to compare the different learning strategies used by two legged robot-platforms in practice. LittleDog by Kalakrishnan et al. [5] and Messor as described by Belter et al. [3] were chosen as they are similar in many respects, but differ in their learning mechanisms. To limit the scope of the paper and concentrate on these essential differences, the focus here lies with the foothold selection methods employed by the platforms. These selections are most important to guarantee that the robot achieves a stable and quick walk across unknown terrain and therefore need to be very flexible while still offering good balance control.

Section 2 gives an overview of the standard control structures of walking robots. This is the followed by Sections 3 and 4, in which the two robot platforms and their foothold selection mechanisms are introduced, along with the corresponding evaluation results. Section 5 then offers a direct comparison of the relevant points concerning learning control and gives some ideas for future work.

2 The basic control architecture

As with all complex systems, it is necessary to gain at least a broad overview over its functionality to be able to understand the details of different aspects later on. In order to do so, the basic control structures of a walking robot will be introduced in this section, to show in which context the foothold selection algorithms are running.

Figure 2 shows the basic process of having a legged robot move to a given goal position and orientation. The figure is based on the architecture designed by Kalakrishnan et al. in [5] and [4], but generalized to encompass other approaches and robots as well. Depending on the implementation, some steps may be performed offline on external systems or in a preparation phase before the actual run of the robot.

Starting with a given goal position and orientation of the robot, usually a general approximate path is planned using the current global terrain map and some general reward function of the terrain. Based on this approximate body path and the current position along it, the footstep planner decides on the next footholds to be used, depending on a previously, usually offline-learned, foothold ranking function. With the next footholds chosen, the pose finder calculates the optimal positioning of the robot for these footholds, so that the best possible maneuverability and stability is ensured for further steps. This positioning includes mainly deciding on the optimal angles for the different leg-joints.

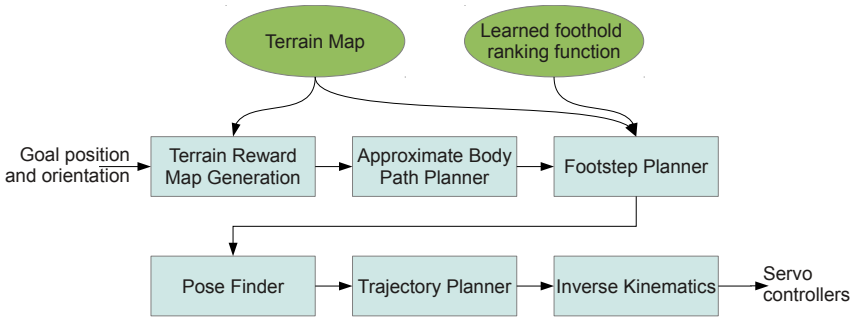

Figure 2: Basic architecture of a walking robot's motion planning system

The trajectory planner then decides on the best way to achieve this position, once again taking into account stability constraints and smoothness of the resulting motion. The resulting trajectory must follow the original approximate trajectory as closely as possible while keeping the robot stable at the same time. It is then encoded in commands for the servo motors controlling the robot using inverse kinematics.

While machine learning techniques may be used in some or all of these steps to gain a better performance, this paper concentrates only on the learning involved in the foothold selection for a single step forward, as was already discussed in Section 1.

3 Practical Application on LittleDog robot

Figure 3: The LittleDog quadruped robot [5]

The work by Kalakrishnan et al. [4] [5] was done in the context of the DARPA Learning Locomotion project which has the goal of improving legged robot locomotion in terms of both speed and robustness. The LittleDog robot platform as shown in Figure 3 is a four-legged robot with three degrees of freedom (DOFs) per leg.

The approach is based on quasi-static locomotion, meaning that the center of gravity or the zero moment point is always kept within the support polygon (the polygon spanned by the footholds) to keep the robot from falling over. In addition to that constraint, a good foothold must give the optimal trade-off between minimizing slippage and maximizing the forward progress. Additionally, and most importantly, it must keep the robot from colliding with the environment.

3.1 Learning features directly

Each potential foothold is described by a feature vector $\mathbf{x}_i$ which contains both information on the terrain at the foothold and the robot's pose after choosing it. The reward function to rank all potential footholds based on these features and select the best possible choice is defined as

$$R(\mathbf{x}_i) = \mathbf{w}^T \mathbf{x}_i.$$

To learn this reward function, Kalakrishnan et al. chose to use *expert demonstrations* as the basic approach. A human expert inspects logs of different runs the robot made and marks the correct foothold placements as opposed to the ones the robot actually selected. These corrections implicitly state that the manually chosen foothold with feature vector $\mathbf{x}_c$ is better than the one for all other possibilities:

$$\mathbf{w}^T \mathbf{x}_c > \mathbf{w}^T \mathbf{x}_i \;\; \forall i \in \mathcal{F}, i \neq c$$

This equation is equivalent to the following one:

$$\mathbf{x}_c - \mathbf{x}_i > 0 \;\; \forall i \in \mathcal{F}, i \neq c$$

Based on this formulation it obviously follows that the optimal weights $\mathbf{w}$ can be obtained by running a linear classifier (e.g. a Support Vector Machine or Logistic Regression) on the difference vectors $(\mathbf{x}_c - \mathbf{x}_i)$.

The main problem of this approach is the linear dependence of the rewards to the features. Only fairly simple decisions can be modelled this way without manually adding features for every special case.

3.2 Learning with terrain templates

In order to overcome this limitation, Kalakrishnan et al. propose to use a library of learned terrain templates with matching rewards. A candidate foothold is then associated with the closest matching template in the library. Again, manual creation of such a library would be infeasible, so the template extraction is done through learning by expert demonstration. For each demonstrated foothold, a set of templates on multiple scales is extracted (see Figure 4), giving information both on terrain details and a broader view of the foothold surroundings. One single template is simply a height-map of the area surrounding the potential foothold. This selection method yields a library of templates that is too large for online classification and that would make the learning algorithm prone to overfitting.

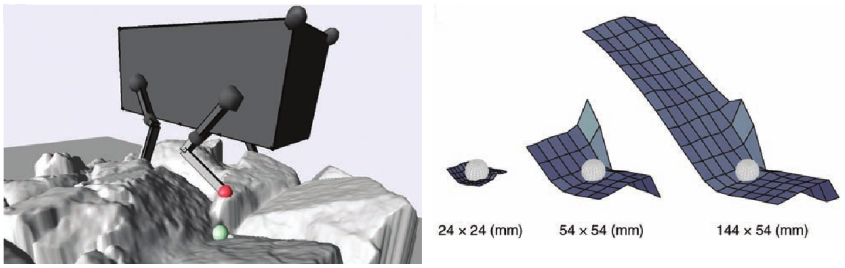

Figure 4: Expert-selected vs. robot-chosen foothold *(left)* and terrain templates at different scales *(right)*. Figure taken from [4].

This problem can be overcome using the following learning algorithm: The feature vector $\mathbf{x}_i$ of a foothold now consists of a similarity measure – Kalakrishnan et al. use a radial basis function – for each template in the library. To calculate the weights and create a sparser set of templates l_1-regularized logistic regression (LR) on the vector $(\mathbf{x}_c - \mathbf{x}_i)$ is used. The details of this particular LR algorithm can be found in [6]. In this case, it is equivalent to the minimization of the following function:

$$J = \sum_{i=1}^{m} -\log\left(\frac{1}{1+\exp(-\mathbf{w}^T \mathbf{x}_i y_i)}\right) + \lambda \|\mathbf{w}\|_1$$

In this case, m is the number of training examples, $y_i \in \{-1, 1\}$ is the label of the i-th input vector and λ is a regularization parameter. The second term in this equation is a regularization term promoting sparsity of the weight vector $\mathbf{w}$. All templates t_i for which $w_i = 0$, can be removed from the library.

3.3 Results of experimental evaluation

For their experimental evaluation, Kalakrishnan et al. chose different terrain modules (e.g. flat modules, steps, rocks, etc.) the robot had to traverse. These were scanned beforehand by a 3D laser scanner. This 3D model in combination with reflective markers and a motion capture system gave the robot complete information on its position and surroundings. The advantage of this setup is, that the main focus could be held on the path planning and control instead of concentrating on information gathering and interpolation of unknown terrain.

The system learned from around 100 expert demonstrated footholds and was then sent to perform 21 test runs over a terrain module not previously seen by it. A run was said to be successful if the robot reached the goal position. Furthermore, slippage was measured, defined as the distance between the foot position when it first touched the ground and the position where it stood before that foot made its next step.

The robot was tested with three different feature sets: Using classic terrain features only (such as slope, curvature and standard deviation), using templates only and using a combination of both. It was found that, while the template-only approach led to a big improvement compared to the approach using only terrain-features, the combination of both beat both others by a wide margin. Using both together yielded a success rate of 100% and an average slip of 17.3 mm $\pm$ 13.4 mm (compared to 13.4 mm $\pm$ 0.4 mm for flat terrain).

4 Practical Application on six-legged robot "Messor"

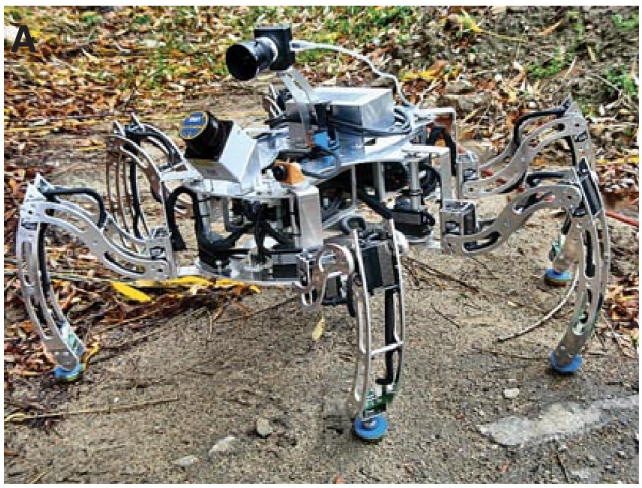

Figure 5: "Messor", a six-legged robot platform [3]

Another approach to this same problem of foothold selection was taken by Belter et al. [3]. Their robot "Messor" is shown in Figure 4. In contrast to Kalakrishnan's LittleDog, Messor is not dependent on perfect maps and global positioning information and uses a different technique for learning footholds that will be described in the following.

Messor applies unsupervised learning in realistic simulations to decide on the best footholds. No external stimuli are needed for this algorithm. Similar to LittleDog, Messor uses slippage as an important optimization criterion, though it is defined a bit differently: Since its gait is based on the tripod nominal gait (meaning at every point in time there are at least three legs on the ground and three are moved forward in parallel), the slip of the ith foot is defined as $r_i = d_i / \|\mathbf{t}_i\|$, where d_i is the distance between the positions at the beginning and the end of the stance phase. $\|\mathbf{t}_i\|$ is a normalization factor, relating the slippage to the actual motion made by the robot – large movements are allowed a larger amount of slippage.

To describe the terrain around a potential foothold, elevation maps created from laser scanner data are used. For each cell on the elevation grid two coefficients K_1 and K_2 are calculated which are

defined as

$$K_1(i,j) = \sum_{k=-1}^{1} \sum_{l=-1}^{1} (z_{i,j} - z_{i+k,j+l})$$

and

$$K_2(i,j) = \sum_{k=-1}^{1} \sum_{l=-1}^{1} \|z_{i,j} - z_{i+k,j+l}\|$$

respectively. In these equations $z_{i,j}$ stands for the height of grid-cell (i,j). While K_1 gives information on whether the cell is a top-edge or a hole, K_2 differentiates between flat terrain and a constant slope. Each of these coefficients by itself does not give enough data concerning the type of the terrain, but together, those two coefficients hold the necessary information of the basic terrain structure at a certain grid-cell.

4.1 Learning footholds in simulation

Learning the footholds in simulation instead of using measurements from the real robot has two main reasons: First of all it is impossible to take exact measurements of foot slippages without expensive motion capture systems and secondly, there is a certain danger to the robot, should it slip too much and fall. For this reason Belter et al. present a simulator based on a learned robot model. To minimize the "reality gap" – the difference between the reactions of the simulated and the real robot when acting on the same control input – they used evolutionary algorithms to learn the best fitting model parameters. This system was first described by them in [2] using another hexapod robot "Ragno". It was then adapted to work for Messor as well.

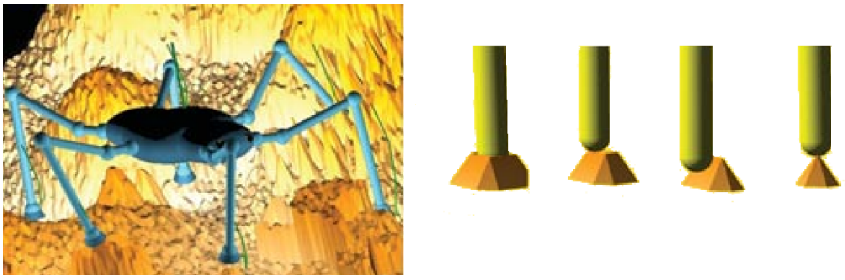

Figure 6: Messor in simulation *(left)* and examples of manually created terrain templates *(right)*. Figure taken from [3].

Having acquired a suitable robot model, two different ways of gathering data were examined: In the first one, the robot selects its footholds randomly and measures the slippage afterwards. The corresponding foothold coefficients and slippages are then saved for further processing afterwards. Somewhat similar to Kalakrishnan's approach, the second acquisition mode uses a set of seven manually prepared ground templates (basically just a fixed set of K_1 and K_2 values, see Figure 6). After placing each robot foot on the same template, again the slippage is registered and saved. To gather a wider range of data, the height of the templates is changed and they are rotated by up to 45. Furthermore some Gaussian noise is added to the templates' z coordinates to add to the diversity of the input data.

4.2 Creating a decision surface

Until now, the gathered data is basically a point-cloud with (K_1, K_2, r) as its x-, y- and z-axis. It is beneficial for further processing to discretize this point-cloud to a regular (K_1, K_2)-grid. The r values of cells containing multiple data points are computed as the mean of those points. To select the most useful ground-points, the following polynomial must be minimized:

$$P(K_1, K_2) = \sum_{q=0}^{m} c_q \cdot \exp\left(\sum_{i=1}^{2} \lambda_{i,q}(K_i - a_{i,q})^2\right)$$

Here, c_q are coefficients computed from the ground-points and slippages. $\lambda_{i,q}$ and $a_{i,q}$ are coefficients defining the Gaussian functions used to approximate the polynomial. For more information on this equation please refer to [3] as the details are beyond the scope of this paper. Since there is no analytical method to optimize the polynomial, Belter et al. employ a population level-based evolutionary algorithm [1] to find acceptable values. In this case, a single gene represents exactly one of the Gaussian functions. During a mutation, all parameters of that gene are changed. According to the learned decision surface, neither peaks nor deep holes are useful as footholds, but small depressions and flat surfaces are usable.

4.3 Evaluation on the real robot

In a first test the robot had to cross a rocky terrain mock-up, starting with a complete elevation map of the area (gained during earlier mapping experiments). The first two runs in this test were conducted using nominal foot positioning (simply move the legs according to the nominal gait) and the new foothold selection algorithm respectively. Comparing the results, it is noticeable that, while both lead to a completion of the mission, the new algorithms allows the robot to keep its trunk mostly horizontally and follows the given trajectory more closely.

As the software created by Belter et al. does not contain high-level path planning, difficult terrain can only be traversed by giving an acceptable base-path decided on by the operator. Using more difficult unknown terrain and a straight-line path lead to a success rate of only 10%, mostly due to self-localization errors that caused incorrect foot placement. With better user-given trajectories that did not exceed the robot's mechanical specifications, a 60% success rate could be achieved for 10 runs. Again, the failures were all caused by incorrect self-localization.

5 Discussion & Conclusion

Kalakrishnan et al. and Belter et al. presented different approaches towards foothold-selection on the legged robots LittleDog and Messor respectively.

LittleDog, using terrain-templates learned by expert-demonstration, performed generally better during the evaluations than Messor, showing minimal slippage and a success rate of 100% for unknown terrain. While this is an impressive feat, the parameters of the examinations may not be ignored. The problem of self-localization that lead to an increased amount of failure for Messor was not an issue for LittleDog, as it was taken out of the equation by means of an external motion capture system. Furthermore LittleDog depends on exact and complete mapping that is not possible using only on-board sensors without previous information. Another aspect that can mainly be observed from the additional videos offered by both parties, LittleDog moves at a much greater speed and is able to overcome obstacles that are outside its usual operational range.

To summarize, LittleDog performs better in terms of speed and success, but is dependent on external information during the task to succeed. Messor, on the other hand, is able to function autonomously after receiving an initial trajectory. It seems, that combining both approaches, using LittleDog's terrain-templates in conjunction with the local mapping employed by Messor might be a rewarding future research project in order to counter the flaws of each system with the strengths of the other.

References

[1] M. Annunziato and S. Pizzuti. Adaptive parameterization of evolutionary algorithms driven by reproduction and competition. In *In Proceedings of ESIT-2000*, 2000.

[2] D. Belter and P. Skrzypczyński. A biologically inspired approach to feasible gait learning for a hexapod robot. *Int. J. Appl. Math. Comput. Sci.*, 20(1):69–84, March 2010.

[3] D. Belter and P. Skrzypczyński. Rough terrain mapping and classification for foothold selection in a walking robot. *J. Field Robot.*, 28(4):497–528, July 2011.

[4] M. Kalakrishnan, J. Buchli, P. Pastor, M. Mistry, and S. Schaal. Learning, planning, and control for quadruped locomotion over challenging terrain. *Int. J. Rob. Res.*, 30(2):236–258, February 2011.

[5] M. Kalakrishnan, J. Buchli, P. Pastor, and S. Schaal. Learning locomotion over rough terrain using terrain templates. In *Proceedings of the 2009 IEEE/RSJ international conference on Intelligent robots and systems*, IROS'09, pages 167–172, Piscataway, NJ, USA, 2009. IEEE Press.

[6] K. Koh, S.-J. Kim, and S. Boyd. An interior-point method for large-scale l1-regularized logistic regression. *J. Mach. Learn. Res.*, 8:1519–1555, December 2007.

[7] S. Schaal and C. Atkeson. Learning control in robotics. *Robotics Automation Magazine, IEEE*, 17(2):20 –29, june 2010.