2008 IEEE/RSJ International Conference on Intelligent Robots and Systems
Acropolis Convention Center
Nice, France, Sept, 22-26, 2008

Learning Predictive Terrain Models for Legged Robot Locomotion

Christian Plagemann Sebastian Mischke Sam Prentice
Kristian Kersting Nicholas Roy Wolfram Burgard

Abstract— **Legged robots require accurate models of their environment in order to plan and execute paths. We present a probabilistic technique based on Gaussian processes that allows terrain models to be learned and updated efficiently using sparse approximation techniques. The major benefit of our terrain model is its ability to predict elevations at unseen locations more reliably than alternative approaches, while it also yields estimates of the uncertainty in the prediction. In particular, our nonstationary Gaussian process model adapts its covariance to the situation at hand, allowing more accurate inference of terrain height at points that have not been observed directly. We show how a conventional motion planner can use the learned terrain model to plan a path to a goal location, using a terrain-specific cost model to accept or reject candidate footholds. In experiments with a real quadruped robot equipped with a laser range finder, we demonstrate the usefulness of our approach and discuss its benefits compared to simpler terrain models such as elevations grids.**

I. Introduction

The advantages of using legged robots over traditional wheeled robots are the ability to move in rough and unstructured terrain and to step over obstacles. Without accurate knowledge of the terrain, these advantages cannot be realized as standard motion planners require a model of terrain height for computing stability and avoiding collisions. However, acquiring and representing models of rough terrain is a challenging task. First of all, terrains are defined over a continuous space such that the space of all models has in principle infinitely many dimensions. Discretizing this continuous space either results in models of enormous size or in a loss of information that in turn may lead to the selection of statically unstable and kinematically infeasible configurations of the robot. Furthermore, we must rely on the robot's noisy sensors to gather information about the world, which requires statistical inference in the continuous and high-dimensional space of the model. Finally and most importantly, we wish to be able to deal with a varying data density, to balance smooth inference of the terrain against the preservation of discontinuities, and to be able to make sensible predictions about unseen parts of the terrain.

In this paper, we present a novel, probabilistic terrain modeling approach based on a Gaussian process (GP) formulation that adapts its generalization behavior to local structure in the terrain and that can be learned and updated efficiently using a decomposition of the model into smaller, overlapping sub-models. As a result, our model is substantially more efficient and accurate than those based on standard Gaussian processes and also outperforms grid-based approaches to elevation mapping in several aspects.

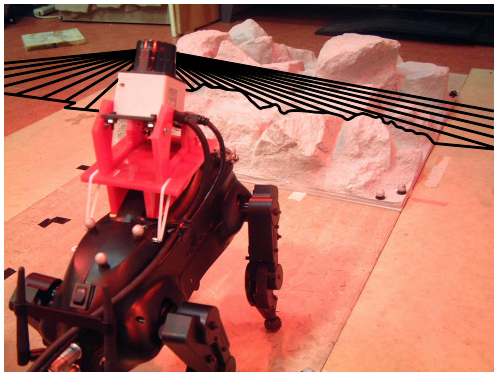


Fig. 1. Our quadruped robot is equipped with a laser sensor in order to observe its local surrounding. By bending its leg joints, it is able to acquire dense 3D scans of the environment.

As a second contribution of this paper, we apply the novel terrain model to the task of terrain mapping with a quadruped robot, specifically the Boston Dynamics LittleDog. In the situation depicted in Fig. 1, our robot, which is equipped with a laser range finder, faces the task of planning and executing a path over rough terrain. It is geometrically impossible for the dog to directly sense every point in the terrain from a single pose, since the sensor has a limited field of view and the terrain self-occludes in multiple locations. The ability to predict terrain elevations at unseen locations, however, is highly beneficial in planning. Although it is difficult to express *a priori* how the local structure can predict the unobserved terrain heights, we will demonstrate in this paper that this relationship can be learned from experience. Finally, we also show how our locally adaptive GP model allows us to select safe foothold locations and to plan a path to a goal location. The terrain model we present in this paper was implemented and tested with a real robot. Fig. 2 shows screen-shots of our controller application in some example situations, i.e., during scan acquisition, terrain adaptation, and path planning.

We proceed as follows. We begin by discussing related work in Section II. We formalize terrain modeling as a nonparametric regression problem in Section III and show how to automatically generate and execute locomotion plans for a quadruped robot in Section IV. Finally, we present our empirical evaluation on a real quadruped robot equipped with

C. Plagemann, S. Mischke, and W. Burgard are with the University of Freiburg, Department of Computer Science, Georges-Koehler-Allee 79, 79110 Freiburg, Germany – {plagem,mischkes,burgard}@informatik.uni-freiburg.de

S. Prentice and N. Roy are with the Massachusetts Institute of Technology, Computer Science and Artificial Intelligence Lab, Cambridge, USA – {prentice,nickroy}@mit.edu

K. Kersting is with the Fraunhofer Institute IAIS, Sankt Augustin, Germany – kristian.kersting@iais.fraunhofer.de

978-1-4244-2058-2/08/$25.00 ©2008 IEEE.

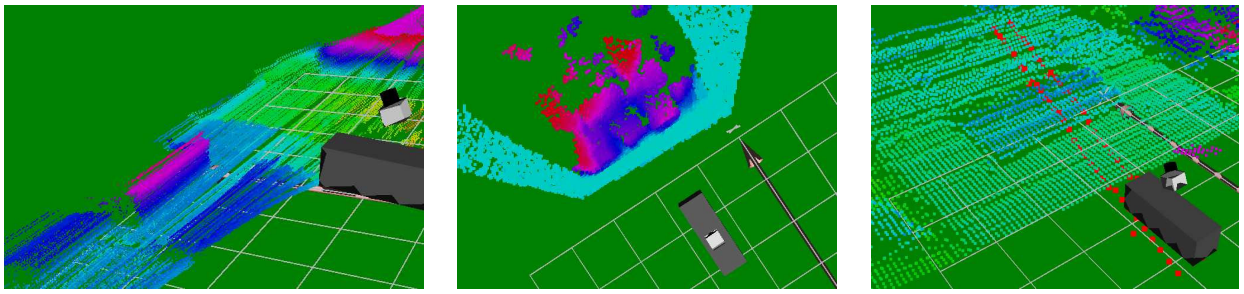


Fig. 2. Online visualization using the controller application of our implemented system in three different scenarios. The robot bends its legs to acquire a 3D scan (left), learns a probabilistic terrain regression model (middle), and then plans a path to a goal location (right). The terrain elevations are color coded, ranging from green/blue (lowest) to pink/red (highest).

a laser range finder in Section V and compare our model to popular grid-based alternatives.

II. Related Work

Terrain modeling and map building are central tasks within robotics, and a broad overview of methods used for modeling terrain data was given by Hugentorp [6]. Elevation maps in particular have been used as an efficient data structure for representing dense terrain data [1], [11] and were later extended to multi-level probabilistic surface maps [19]. Früh *et al.* [4] present an approach to filling local gaps in 3D models based on local linear interpolation. Compared to these approaches, Gaussian process (GP) models [15] have the advantage of not assuming a fixed discretization of the space and of additionally providing predictive uncertainties. The explicit model of uncertainty that a GP provides has led to their successfully application in a wide range of other applications areas such as positioning systems using cellular networks [16].

Within robotics, GP models have recently become popular, for instance for measurement models [2] or model-based failure detection [12], because they naturally deal with noisy measurements, unevenly distributed observations, and fill small gaps in the data with high confidence while assigning higher predictive uncertainty in sparsely sampled areas. Many robotics applications, however, call for non-standard GP models. For instance, in our problem domain the terrain model should preserve structural elements such as edges and corners as they are important features for path planning. We follow Lang *et al.* [8] and model the terrain using a GP with a nonstationary covariance function originally proposed by Paciorek and Schervish [10]. If not specifically addressed, the nonparametric nature of this approach causes computational problems for large terrains, due to an unfavorable N^3 scaling for training, where N is the number of observations. To overcome this problem, we propose to use an ensemble of GPs, where every GP is assigned to a specific region, an idea akin to GP mixture models such as Williams' [20]. Rasmussen and Ghahramani [14] extend the ideas of Tresp [18] and present an "infinite mixture of experts model" where the individual experts are different Gaussian process models. Cornford *et al.* [3] model straight discontinuities in wind fields by placing auxiliary GPs along the edge on both sides of the discontinuity. These are then used to learn GPs representing the process on either side of the discontinuity.

III. Terrain Mapping as a Regression Problem

Traversable surfaces can be characterized by a function $f : \mathbb{R}^2 \to \mathbb{R}, f(\mathbf{x}) = y$, where $\mathbf{x}$ indexes a location in the 2D plane and y denotes the corresponding terrain elevation in the direction opposing the gravity vector. Elevation grids are a popular way of representing such functions and learning them from a set of elevation samples $\mathcal{D} = \{\mathbf{x}_i, y_i\}_{i=1}^n$. In the grid formulation, the space of locations $\mathbf{x}$ is discretized, such that each grid cell is assigned a constant elevation value [11] or a parametric function $p(y)$ is fitted to the distribution of its elevation values [19]. Gaussian processes (GPs) take a different approach by viewing any finite set of samples y_i from the sought-after distribution as being jointly normally distributed,

$$p(y_1, \ldots, y_n \mid \mathbf{x}_1, \ldots, \mathbf{x}_n) \sim \mathcal{N}(\boldsymbol{\mu}, K) \ , \tag{1}$$

with mean $\boldsymbol{\mu} \in \mathbb{R}^n$ and covariance matrix K. $\boldsymbol{\mu}$ is typically assumed $\mathbf{0}$ and K is specified in terms of a parametric covariance function k and a global noise variance parameter σ_n^2, $K_{ij} = k(\mathbf{x_i}, \mathbf{x_j}) + \sigma_n^2 \delta_{ij}$. The covariance function k represents the prior knowledge about the target distribution and does not depend on the target values y_i of $\mathcal{D}$. A common choice is the squared exponential (SE) covariance function

$$k(\mathbf{x}_i, \mathbf{x}_j) = \sigma_f^2 \exp\left(-\frac{1}{2}\sum_{d=1}^{2}\frac{(\mathbf{x}_{i,d} - \mathbf{x}_{j,d})^2}{\ell_d}\right) \ , \tag{2}$$

where σ_f denotes the amplitude (or signal variance) and ℓ_d is the characteristic lengthscale of dimension d (see [15]). These parameters, along with the global noise variance σ_n^2, are known as the hyperparameters of the process and denoted as $\boldsymbol{\Theta} = (\sigma_f, \boldsymbol{\ell}, \sigma_n)$. Since any set of samples from the process is jointly Gaussian distributed, the prediction of a new target value y^* at a given location $\mathbf{x}^*$ can be performed by conditioning the n+1-dimensional joint Gaussian on the known target values of the training set $\mathcal{D}$. This yields a predictive normal distribution $y^* \sim \mathcal{N}(\mu^*, v^*)$ defined by

$$\mu^* = E(y^*) = \mathbf{k}^T \left(K + \sigma_n^2 I\right)^{-1} \mathbf{y} \ , \tag{3}$$

$$v^* = V(y^*) = k^* + \sigma_n^2 - \mathbf{k}^T \left(K + \sigma_n^2 I\right)^{-1} \mathbf{k} \ , \tag{4}$$

with $K \in \mathbb{R}^{n \times n}$, $K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j)$, $\mathbf{k} \in \mathbb{R}^n$, $k_j = k(\mathbf{x}^*, \mathbf{x}_j)$, $k^* = k(\mathbf{x}^*, \mathbf{x}^*) \in \mathbb{R}$, and the training targets $\mathbf{y} \in \mathbb{R}^n$. To best represent the underlying data, the hyperparameters $\boldsymbol{\Theta}$ need to adapted. The common way of doing this is by maximizing the marginal log likelihood of the training data w.r.t. the hyperparameters.

A. Locally Adaptive Gaussian Processes

A limitation of the standard GP framework is the assumption of constant lengthscales over the whole input space. Intuitively, the lengthscales describe the area in which observations strongly influence each other. For terrain models, one would like to use locally varying lengthscales to account for the different situations. For example, in flat plains, the terrain elevations are strongly correlated over long distances. In contrast, in high-variance, "wiggly" terrain and at discontinuities in the terrain, the terrain elevations are correlated over very short distances, if at all. To address this problem of *nonstationarity*, an extension of the squared exponential (SE) covariance function was proposed by Paciorek and Schervish [10]. It takes the form

$$k(\mathbf{x}_i, \mathbf{x}_i) = |\Sigma_i|^{\frac{1}{4}} |\Sigma_j|^{\frac{1}{4}} \left| \frac{\Sigma_i + \Sigma_j}{2} \right|^{-\frac{1}{2}} \times \exp\left[-(\mathbf{x}_i - \mathbf{x}_j)^T \left(\frac{\Sigma_i + \Sigma_j}{2} \right)^{-1} (\mathbf{x}_i - \mathbf{x}_j) \right]. \quad (5)$$

Each input location $\mathbf{x}'$ is assigned a local Gaussian kernel matrix Σ' and the covariance between two targets y_i and y_j is calculated by averaging between the two local kernels at the input locations $\mathbf{x}_i$ and $\mathbf{x}_j$. In this way, the local characteristics at both locations influence the modeled covariance of the corresponding target values. Lang *et al.* [8] applied this model to the terrain regression problem and derived an iterative adaptation procedure that does not require Markov chain Monte Carlo integration as the original approach did. In order to further increase the efficiency of the model and to make it conceptually simpler, we derived the following non-iterative model.

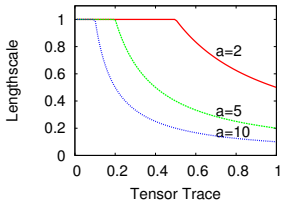
Following Lang *et al.* [8], we adapt the the local kernels Σ_i using terrain gradient information. For an input location $\mathbf{x}_i$, we estimate the gradient $(\nabla y)_i$ from elevation observations in the local neighborhood. We then calculate the trace of the elevation structure tensor

$$T(\mathbf{x}_i) = \text{trace}\left(\overline{(\nabla y)_i (\nabla y)_i^T} \right) \quad (6)$$

to yield a single scalar representation of the terrain's slope. Here, $\overline{\cdot}$ denotes the locally weighted averaging operator. To achieve the desired smoothing behavior, we calculate the *local* lengthscale

$$l(\mathbf{x}_i) = \begin{cases} a \cdot T(\mathbf{x}_i)^{-1} & \text{if } a \cdot T(\mathbf{x}_i)^{-1} < l_{max} \\ l_{max} & \text{else} \end{cases} \quad (7)$$

in order to yield short lengthscales in high variance terrain and long lengthscales in flat parts. $l(\mathbf{x}_i)$ is bounded by l_{max} to prevent lengthscales from going to infinity in large, flat regions. The scale parameter a is learned in parallel to the search for the GP's hyperparameters. We visualize $l(\mathbf{x}_i)$ in Fig. 3 for different parameter settings. The local kernel Σ_i is then simply an isotropic kernel $\Sigma_i = l^2(\mathbf{x}_i) \cdot I$ with eigenvalues $l(\mathbf{x}_i)$.



Missing data	WA	HGP
15%	0.437	0.104
30%	0.531	0.168
50%	0.735	0.406

Fig. 3. Left: The parametric function $l(\mathbf{x}_i)$ links terrain slope to lengthscales. Right table: Placing a *hyper GP* (HGP) over the latent lengthscales reduces the mean squared error (MSE) of predictions w.r.t. to *weighted averaging* (WA) for different levels of missing data.

To be able to make elevation predictions at arbitrary locations, we need to evaluate the covariance function at arbitrary locations and, thus, need to have local kernels at any point in the input space. We are only able to calculate gradients and kernels directly where we have sufficient elevation observations in the local neighborhood. Whereas Lang *et al.* [8] use simple weighted averaging to calculate kernels in regions with few or no observations, we propose to instead put another GP prior on the local kernels' parameters. We call this a *hyper GP*, since its predictions are used as hyper parameters for the GP that models the elevations.

The hyper GP represents the function $\mathbf{x} \rightarrow l(\mathbf{x})$ for the lengthscales of the main GP. As the lengthscales have to be positive, we transform its training data into log space. The hyper GP itself uses an isotropic lengthscale l_h, which can be treated as an additional hyperparameter of the model. At inference time, we calculate the most likely lengthscale given by the mean prediction of the hyper GP and use the resulting kernels for elevation predictions of the elevation GP. The improvement in elevation prediction using the hyper GP (HGP) with respect to the weighted averaging (WA) approach of Lang *et al.* [8] is shown in the table of Fig. 3. We give the results for three different percentages of points removed from the training set.

B. Model Tiling

The cubic time complexity of standard GPs limits their applicability to large data sets. In terrain modeling tasks, the training data grows as new parts of the terrain are explored. With only a few laser scans, the training points are in tens of thousands and regular GPs would spent days on fitting the regression function.

We propose to use an ensemble of overlapping GPs, where every sub-model is assigned to a specific region in the input space. For most of the covariance functions relevant in practice, the covariance between two points decreases drastically with their distance (e.g., exponentially for SE). The lengthscale parameter of the covariance function is simply a factor scaling this decay, which does not influence

the asymptotic behavior. Consequently, the decrease in covariance is asymptotically the same for our nonstationary covariance function, where lengthscales vary smoothly and are ultimately bounded by a maximum value. If a point lies sufficiently far from a region in input space, it has virtually no influence on the regression result within this region. Thus, we propose to split the input space into rectangular segments, and to assign an individual GP model to each of the segments. This sub-model is then only provided with observations from within its segment. This idea is similar to the approximation, which considers the full dependencies between observations if they belong to the same segment, but applies approximations for longer distances [17]. To avoid problems at segment boundaries, we arrange the segments in an overlapping fashion and only use the center parts of the segments for the final prediction. In this way, we overcome the problem that predictions close to segment borders have unreasonably high variance [17].

Concretely, for a prediction at input location $\mathbf{x}$, we first determine the GP segment which we consider most likely to have the best approximation for $\mathbf{x}$, i.e., the segment which has a center that is the shortest Euclidian distance to $\mathbf{x}$. Given this hard assignment to segments, the resulting function is no longer continuous in general. We have, however, not observed notable problems in practice, which might be related to this approximation. In order to evaluate the gain in runtime performance, let us assume that

a) every segment contains at most a percentage c of all training data n, and
b) segments overlap by a percentage v of their inputs.

Every segment then uniquely covers $cn - cnv = cn(1-v)$ of the training data, which makes it necessary to use $\frac{n}{cn(1-v)} = (c(1-v))^{-1}$ segments to cover the whole input space. The original GP training time of $O(n^3)$ can then be expressed as $O((c(1-v))^{-1} \cdot (cn)^3) = O(c^2(1-v) \cdot n^3)$. If we keep v constant, and scale c anti-proportionally to n (which corresponds to a constant segment size), the training time becomes linear in n.

In the experiments documented in Section V, we specifically evaluated the benefits of our model tiling strategy, as well as the advantages that the locally adapted GP has w.r.t. the standard GP model and grid-based approaches. Before doing so, we describe how useful trajectories for a quadruped robot can be generated using the predictive elevation model introduced in this section.

IV. Plan Generation

We can now use the learned terrain models to plan a path for the robot that is collision-free and statically stable. Our overall planning approach is an adaptation of the conventional probabilistic roadmap algorithm [7]. Our simplified model of the quadruped robot is a body with four two-link legs and point feet. The planning algorithm is a search for motions of single legs from static stance to static stance that maintain static stability over uneven terrain. We first randomly sample a set of potential footholds across the terrain, which are used to generate a graph of potential

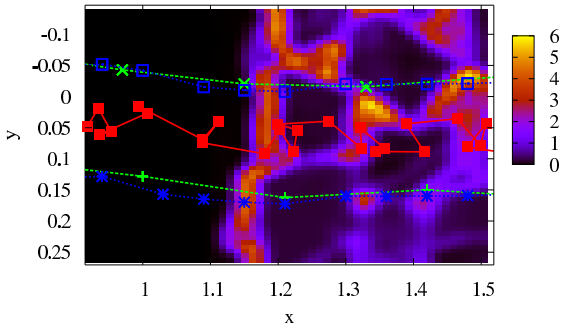


Fig. 4. Parts of a plan generated by our algorithm including the underlying cost function, which depends on the terrain gradient and the uncertainty about elevation predictions estimated by the Gaussian process model. The red line (filled boxes) depicts the trajectory of the center of body, the other lines (stars and empty boxes) visualize feet motions. The cost function is color coded ranging from black (little costs) to yellow/light-gray (high costs). Axis dimensions are given in meters.

"stances", that is, statically stable and kinematically feasible positions of the robot. Graph search is then used to find a sequence of stances from the start to the goal; the sequence of stances can then be converted to series of planned joint angles.

A. Sampling Footholds

Let us assume that the planning problem is to find a motion plan that is essentially a futtock (midline) motion across the uneven terrain from the start position to the goal. This assumption will allow us to simplify the sampling to examining potential footholds around the straight line to the goal, selecting footholds $\phi = (x, y, z)$ according to some regular discretization around the line of intended motion, e.g., see the right diagram of Fig. 2 for the simplest case of equidistantly sampled footholds around a straight line. We do this without loss of generality; we can easily support more complex scenarios by choosing different sampling strategies[1].

Each sampled foothold is evaluated with respect to a cost function and rejected if the expected cost is above some threshold. The cost function may consist of many features including elevation, roughness, but in this work we considered only terrain gradient (i.e., slope) and the uncertainty in the terrain model (i.e., the Gaussian process predictive uncertainty) at the sampled foothold ϕ.

B. Stance Graphs

We next generate feasible stances of the robot from the discrete set of footholds. A stance is an assignment of each foot i to a foothold, $\phi_i = \mathbf{x}$, such that it is kinematically feasible for the robot to place its feet at each of the four footholds and remain statically stable. Note that determining whether a stance is feasible or not is not directly computable from a set of foot positions because the feet do not provide

[1]The sampling problem is outside the scope of this paper but has been discussed in [5] and others.

a unique description of the pose of the robot. The robot has 18 degrees of freedom total: 6 degrees of freedom of the center of the body $(x, y, z, roll, pitch, yaw)$ and the 3 joints $(hip_x, hip_y, knee)$ in each leg. Under the assumption that the positions of the feet are fixed, the feet constitute 12 constraints, leaving 6 unconstrained degrees of freedom, corresponding to the position of the center of body. A stance s_i is therefore an assignment of feet to footholds $\phi_{1..4}$ and a selection of a center of body position ξ.

Given an assignment of the center of the body position for a set of foot positions, the known kinematics can be used to recover the joint angles of the legs and determine if the pose is consistent with the dimensions of the leg links and the limits on the joint angles. Given knowledge of the joint angles and that the stance is kinematically feasible, the center of mass can then be determined; if the projection of the center of mass onto the ground plane lies outside the support polygon (the convex hull of the four feet on the ground plane), then the stance is not statically stable and the robot will fall.

In assigning the position of the center of body for a given set of foot positions, we would ideally choose a center of body that provides static stability. Unfortunately, no closed form solution exists for finding a feasible and stable center of body, and the problem is in general non-convex. We therefore use a heuristic search strategy around the centroid of the support polygon. If none of the sampled centers of body provide a kinematically feasible and stable solution to the robot position given the foot positions, then the foot positions are rejected as an infeasible stance.

The feasible stances constitute nodes in a stance graph, to which we then add edges between pairs of stances s_i and s_j when a feasible motion exists to transform the robot from the start stance s_i (foot positions and center of body) to the end stance s_j (see Fig. 5). This problem is also underdetermined, in that an arbitrarily complex motion may be required to move from one stance to another. We therefore simplify this problem to consider motions consisting of a stance phase, during which the dog shifts its center of body to remain stable while stepping, and a foot swing phase during which a foot is moved from one foothold to another.

Once the stance graph has been built, we use standard breadth-first search to find the shortest feasible sequence of stances from the start stance to a goal stance that gives a center of body position with some ϵ of the desired goal position, in practice, combining the search process with the stance graph generation. Additionally, we add a gait-order constraint, so that the plan must consist of a well-formed gait in which foot i is followed by foot $i + 1 \mod 4$. By augmenting each stance variable with an additional foot-ordering variable ψ, this gait-ordering constraint dramatically improved the planning speed. Finally, we also use a hash table to prune the search, such that if two different routes are found to the same stance node s, then the search along the longer path is terminated. The full planning algorithm is given in Algorithm 1. The hash table is omitted for space reasons.

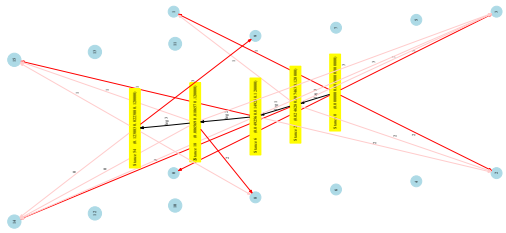


Fig. 5. An example stance graph generated from the set of foothold locations. Each stance node (yellow) is connected to four foothold nodes (blue). Each two connected stance nodes have three stance legs in common.

Algorithm 1 The Planning Process.

Require: Terrain model, start stance s_0 and goal $\mathbf{x}_g$.
  Sample footholds Φ using terrain model
  Initialize $Q \leftarrow s_0$
  while Q is not empty **do**
    $s \leftarrow$ pop Q
    for all ϕ **do**
      $s' \leftarrow s$
      Update position of foot to move, $\phi(\psi(s')) \leftarrow \phi$ in s'
      Update foot to move, $\psi(s') \leftarrow \psi(s') + 1 \mod 4$
      Search for new center of body position $\xi(s')$
      if $||\xi(s') - \mathbf{x}_g|| < \epsilon$ **then**
        return Parents $[s']$.
      end if
      if $\xi(s')$ exists **then**
        Set parent, $\pi[s'] = s$
        Push $Q \leftarrow s'$
      end if
    end for
  end while
  return nil

V. Experiments

The goal of the experimental evaluation is to demonstrate the usefulness of the predictive terrain model introduced in Section III for path planning in real environments and to show that our adapted Gaussian process model is more accurate than conceptually simpler approaches such as standard GPs or elevation grid maps. We additionally analyzed the benefits of our tiling approach introduced in Section III-B with respect to runtime and accuracy performance.

A. Adapting to Local Terrain Structure

We evaluated the performance of the GP terrain regression using the standard squared exponential (SE) covariance function against our nonstationary covariance function with local lengthscale adaption. Our evaluation terrain was an excerpt from the rocky terrain depicted in the left diagram of Fig. 6, in which we simulated 2,500 laser observations from a single viewpoint. We uniformly selected 4,350 points from the true terrain for evaluation. We then conducted Monte-Carlo search in the parameter space of the covariance functions and on the parameters of the adaption procedure. In a preliminary run over 34,000 configurations, we determined general ranges for the parameters and in a secondary search, we evaluated 10,000 configurations in the predetermined ranges. A scatter plot of the best results is given in the

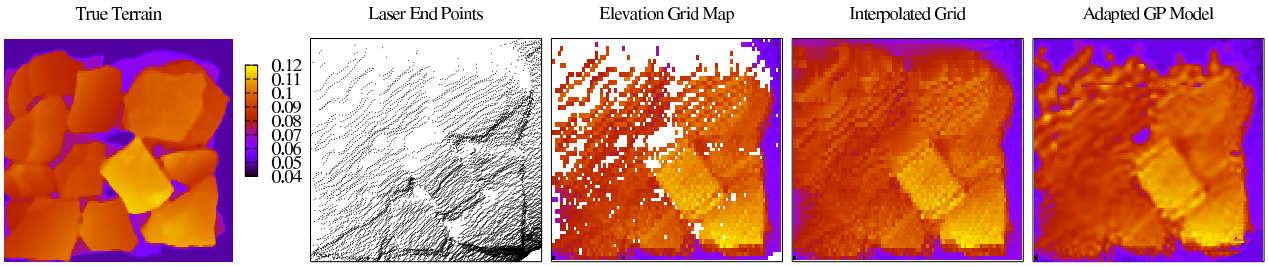


Fig. 6. Mapping results on a rocky, real terrain of dimensions 0.6m × 0.6m with a maximum elevation of approximately 7cm (color-coded as yellow) above ground level. We give the true elevation values (left) and the set of laser end points (second plot) recorded by the robot standing on the bottom right corner w.r.t. the diagram and performing a tilt motion. The next two diagrams depict the elevation grid model and its bilinear completion. The rightmost diagram gives the mean predictions of our adapted GP model.

right diagram of Fig. 8. It can be seen from the diagram that the nonstationary covariance function is able to achieve both better mean squared error (MSE) of predictions w.r.t. the ground truth and also better negative log predictive likelihood (NLPD) that also takes into account the predicted variances. Fig. 8 shows a cut through the respective regression surfaces of the stationary (left panel) and nonstationary (middle panel) models. It can clearly be seen that the stationary GP model (left diagram) needs to select an extremely small lengthscale parameter in order to represent the sharp edges. It is thus not able to smooth flatter and more sparsely sampled regions. In contrast, the nonstationary covariance function (middle diagram) decreases the lengthscale only at the cliffs and where no gradients are observed. It is thus able to account for the highly varying parts at the first ascent and for the higher uncertainties in the occluded parts, while still smoothing in the flat regions.

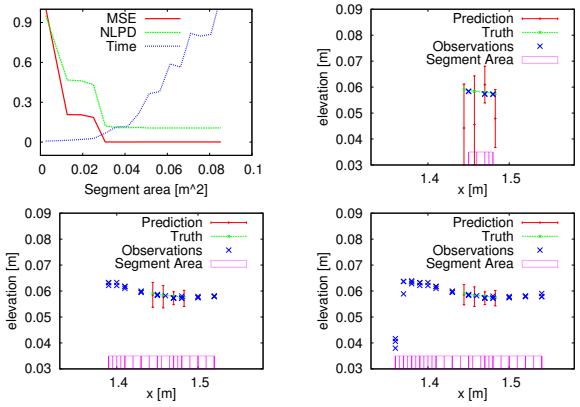


Fig. 7. Mean squared error (MSE), negative log predictive likelihood (NLPD) of the inner $0.0025m^2$ of the segment and train time (relative plot, scale omitted), for different segment areas (left). Values are linearly scaled to fit in plot. Plots of prediction for segment sizes $0.0025m^2$, $0.03m^2$, and $0.056m^2$.

B. Splitting the Terrain into Overlapping Sub-Models

We evaluated the benefits of segmenting the input space in overlapping tiles. To do so, we applied different tile sizes to the terrain model analyzed in the previous experiment. We measured the prediction accuracy in the innermost $0.0025m^2$ of a tile while linearly increasing the tile area, and thereby also the amount of training data of the associated GP. The upper left diagram of Fig. 7 shows that with an increasing tile size, both MSE and NLPD quickly decrease and almost converge as the area reaches $0.003m^2$. The runtime, however, continues to grow cubically with the segment size beyond this point. The remaining three diagrams in Fig. 7 give a visual impression of the effects that different tile sizes have on the regression function in its center. Increasing the tile size from $0.03m^2$ (lower left) to $0.056m^2$ (lower right) does not lead to a notable improvement.

The runtime requirements for learning from approx. 100,000 points using our C++ implemenation are in the order of 1.5 sec. with overlapping stationary GPs and 3 sec. with overlapping nonstationary GPs.

C. Mapping Accuracy on Real Terrain

We evaluated our terrain model with a real quadruped robot in a situation similar to the one depicted in Fig. 1 The robot, called *LittleDog*, was developed by Boston Dynamics. We have equipped the dog with a Hokuyo URG laser scanner. A high-resolution motion capture system, the Vicon MX, yields estimates of the robot pose using measurements from reflective markers attached to the robot's body. The laser sensor is mounted to the back of the robot in a 25° angle facing towards the ground, such that (a) terrain elevation measurements can be acquired while executing plans and (b) 3D range scans can be recorded by executing a tilt motion using the front and rear legs. The evaluation in this section concentrates on the question of how accurately the elevation structure of the terrain board (approximately 0.6m × 0.6m) can be recovered from a single such 3D scan.

1) Calibration: Note that our system is not performing simultaneous localization and mapping; we are only examining the terrain inference problem and assume accurate knowledge of the pose of the laser. The specifics of the Vicon system used to track the dog position required us to automatically infer the time offsets between the laser data and pose measurements, and to calculate the 6 degree of freedom transformation between a known position of the dog and the actual laser sensor position at the same time. In order to compute both these calibration quantities, we recorded a

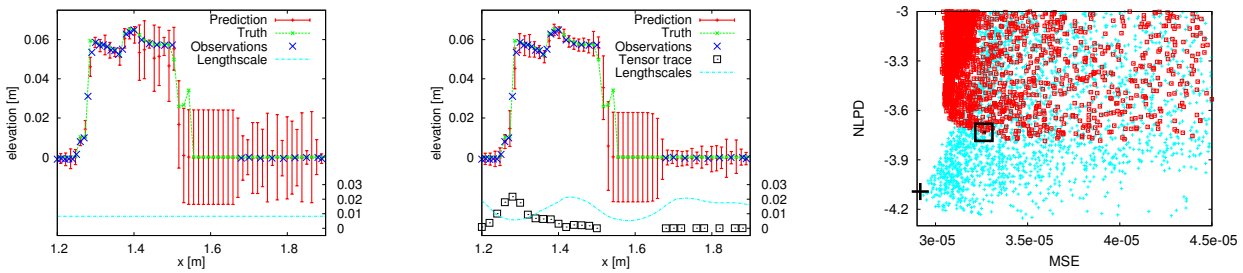


Fig. 8. Regression models learned using the standard GP model (left) and our locally adaptive approach (middle). Here, we visualize a slice plane of the terrain surface (y is fixed, x varies on the horizontal axis in Meters, terrain elevation on the vertical axis). We also give the adapted lengthscales below the curves using a second vertical axis). The right plot visualizes the Monte-Carlo parameter searches for the stationary GP (red boxes) and for the nonstationary GP (bright crosses). The selected optima, which minimize the MSE as well as the NLPD error, are marked by a large black box and by a large black cross respectively.

3D range scan of three orthogonal boards placed in front of the robot (side lengths approximately 1m) and optimized the 7 parameters of the transformation in a sampling-based fashion similar to simulated annealing. Here, random parameter samples are evaluated using a predefined objective function; additionally, a *temperature* variable defines from which area new parameter samples should be drawn around the current optimum. As the objective function, we chose the sum of squared distances of laser end points to the board surfaces. By decreasing the temperature level gradually over time, accurate calibration parameters are typically obtained within 300-800 iterations.

2) Mapping Rough Terrain: We evaluated our adapted GP approach using scans of a rocky terrain surface acquired by the quadruped robot against a known ground-truth model of the terrain acquired using a high-accuracy metrology system. The left diagram of Fig. 6 depicts the true elevation structure of this terrain (see the caption for details). The second diagram gives the raw set of laser endpoints that were acquired by the robot when it executed a tilt motion. It can be clearly seen from the uneven distribution of points that parts of the terrain are not sampled densely due to occlusions and a larger distance to the sensor location (which was towards the bottom right w.r.t. the diagram). An elevation grid map can be built from this training set of surface points by discretizing the x-y space and by fitting 1D Gaussians to the elevation samples falling into the grid cells respectively. The result of this operation is depicted in the middle diagram of Fig. 6.

A standard way of filling gaps in grid maps without altering the known cells is called *bilinear interpolation*, i.e., the extension of linear interpolation to bivariate functions. The result of such an operation applied to an incomplete elevation grid map is depicted in the fourth diagram of Fig. 6, the results obtained by our adapted GP approach are depicted in the rightmost diagram. Here, we plot the mean predictions for terrain elevations. A vertical cut through this surface is given in the middle diagram of Fig. 8. Such a visualization highlights how our model is able to produce uncertainties in the predictions. These uncertainties are utilized in the cost function of the planner described in Section IV to avoid uncertain areas.

We compared the prediction errors of our adapted GP model to the baseline models *Elevation Grid* and *Interpolated Elevation Grid*. In Fig. 9, we give the squared error of elevation predictions averaged over 10,000 samples drawn randomly from the terrain. The error-bars give the standard deviations of the individual sample sets. To assess the influence of grid resolution for the two grid-based models, we tested six different numbers of cells per grid dimension (x-axis). Since the standard elevation grid does not make predictions in occluded or less densely sampled areas, its performance was evaluated on its occupied cells only. It can be seen from the diagram that our adapted GP model is as good as the elevation grid at optimal resolution, although its performance measure additionally includes predictions of unobserved elevations.

D. Path Planning using Real Terrain Data

In order to evaluate the practical usefulness of our approach, we tested our terrain models in combination with the trajectory planner described in Section IV. The experimental setup was to sample 1,000 random starting locations in front of the terrain and to pick goal locations behind it. For each of these location pairs and each of the three alternative terrain models, the planner generated a set of footholds and searched for the best path towards the goal location. We then evaluated (a) the maximal path length that could be constructed given

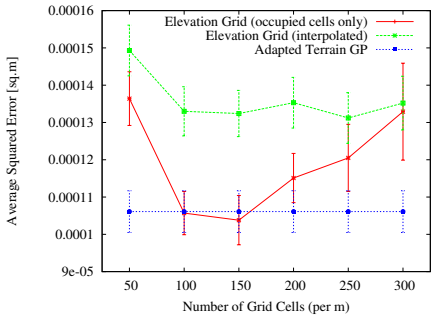


Fig. 9. Errors of elevation predictions by the different models depending on the number of grid cells used. The elevation grid (without interpolation) was evaluated on occupied cells only.

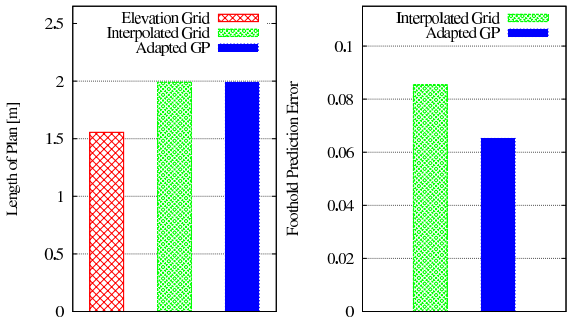


Fig. 10. Evaluation of 1,000 plans generated using the different terrain models. The left bar plot shows the maximal length of generated plans and the right plot gives the mean squared errors (scaled by 10^{-3}) of elevation predictions at the planned footholds.

the kinematic constraints of the robot, and (b) the errors of the elevation predictions at the selected foothold locations. An example plan and the corresponding cost function that was computed from the underlying terrain model are depicted in Fig. 4

Fig. 10 summarizes our results. It can be seen from the left diagram that it was always possible to plan the maximal path of 2 meters using the interpolated grid and the adapted GP model. Using the standard grid, however, the plans never exceeded a length of 1.6 meters, which is not surprising given the large number of unknown cells which prohibit foot placements. As can be seen from the the mean squared error values on the right diagram, the adapted GP model better predicts the true terrain elevations at the chosen foothold locations than the interpolated grid model, which means that there is a lower risk of failure when executing these plans.

VI. Conclusions and Outlook

In this paper, we considered the terrain mapping problem for a legged robot equipped with a laser range finder. We demonstrated the use of a nonparametric Bayesian regression approach, Gaussian processes, for reliably modeling rough terrain. Our extended model balances smoothing against the preservation of structural features and is capable of accurately predicting elevations in the presence of noise even at unobserved locations. These features allow us to plan foot trajectories of a quadruped robot to reach a goal location. We showed in experiments with data acquired using a real robot that this nonparametric terrain modeling approach could infer the terrain more reliably, leading to better planning than grid-based terrain models.

In the future, we plan to apply alternative ways of learning the nonstationary GP model [13] and to evaluate the difference in modeling accuracy. We would also like to address the more general SLAM problem, in which the robot does not assume knowledge of its own position. Future work could also consider a reinforcement learning variant of our foot trajectory planning along the lines of Neumann *et al.* [9] using specific reward functions to learn obstacle avoidance or stable and energy-efficient movements.

Acknowledgments

We would like to thank Tobias Lang for contributing initial source code. This work has partly been supported by the EC under contract numbers FP6-004250-CoSy, by the DFG under contract number SFB/TR-8, and by the BMBF through the DESIRE project. We would also like to thank Russ Tedrake and the DARPA Learning Locomotion project (AFRL contract #FA8650-05-C-7262).

References

[1] J. Bares, M. Hebert, T. Kanade, E. Krotkov, T. Mitchell, R. Simmons, and W. Whittaker. Ambler — an autonomous rover for planetary exploration. 22(6):18–26, June 1989.

[2] A. Brooks, A. Makarenko, and B. Upcroft. Gaussian process models for sensor-centric robot localisation. In *Proc. of the IEEE Int. Conf. on Robotics & Automation (ICRA)*, 2006.

[3] D. Cornford, I. Nabney, and C. Williams. Adding constrained discontinuities to gaussian process models of wind fields. In *Advances in Neural Information Processing Systems 11*. Cambridge, MA, 1999.

[4] C. Früh and A. Zakhor. Data processing algorithms for generating textured 3d building façade meshes from laser scans and camera images. In *3DPVT*, pages 834–849, 2002.

[5] D. Hsu, G. Sánchez-Ante, and Z. Sun. Hybrid PRM sampling with a cost-sensitive adaptive strategy. In *Proc. IEEE Int. Conf. on Robotics & Automation*, pages 3885–3891, 2005.

[6] M. Hufentobler. *Terrain Modelling with Triangle Based Free-Form Surfaces*. PhD thesis, University of Zurich, 2004.

[7] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high dimensional configuration spaces. *IEEE Trans. on Robotics and Automation*, 12(4), 1996.

[8] T. Lang, C. Plagemann, and W. Burgard. Adaptive non-stationary kernel regression for terrain modelling. In *Proc. of the Robotics: Science and Systems Conference (RSS)*, 2007.

[9] G. Neumann, M. Pfeiffer, and Maass W. Efficient continuous-time reinforcement learning with adaptive state graphs. In *Proc. of the European Conference on Machine Learning (ECML)*, 2007.

[10] C. Paciorek and M. Schervish. Nonstationary covariance functions for Gaussian process regression. In *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA, 2004.

[11] P. Pfaff and W. Burgard. An efficient extension of elevation maps for outdoor terrain mapping. In *Proc. of the Int. Conf. on Field and Service Robotics (FSR)*, pages 165–176, Port Douglas, QLD, Australia, 2005.

[12] C. Plagemann, D. Fox, and W. Burgard. Efficient failure detection on mobile robots using particle filters with gaussian process proposals. In *Proc. of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI)*, Hyderabad, India, 2007.

[13] C. Plagemann, K. Kersting, and W. Burgard. Non-stationary gaussian process regression using point estimates of local smoothness. In *Proc. of the European Conference on Machine Learning (ECML)*, Antwerp, Belgium, 2008. To appear.

[14] C. Rasmussen and Z. Ghahramani. Infinite mixtures of Gaussian process experts. In *Advances in Neural Information Processing Systems 14*. MIT Press, Cambridge, MA, 2002.

[15] C. E. Rasmussen and C. K.I. Williams. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. The MIT Press, Cambridge, Massachusetts, 01 2006.

[16] A. Schwaighofer, M. Grigoras, V. Tresp, and C. Hoffmann. A Gaussian process positioning system for cellular networks. In *Proc. of the Conf. on Neural Information Processing Systems (NIPS)*. 2004.

[17] E. Snelson and Z. Ghahramani, editors. *Local and global sparse Gaussian process approximations*, volume 11, 2007.

[18] V. Tresp. Mixtures of gaussian processes. In *Proc. of the Conf. on Neural Information Processing Systems (NIPS)*, 2000.

[19] R. Triebel, P. Pfaff, and W. Burgard. Multi-level surface maps for outdoor terrain mapping and loop closing. In *Proc. of the International Conference on Intelligent Robots and Systems (IROS)*, 2006.

[20] O. Williams. A switched Gaussian process for estimating disparity and segmentation in binocular stereo. In *Proc. of the Conf. on Neural Information Processing Systems (NIPS)*, 2006.