A Gait Library for Rapid Quadruped Locomotion

Sam Schreiber (schreib@stanford.edu)

With advice from Zico Kolter and Professor Andrew Ng

Overview

Cutting-edge robotics is increasingly moving away from simple repetitive tasks on the factory floor and toward complex operations in unstructured natural environments. Such environments present a wide variety of challenges, particularly for conventional modes of robot locomotion: only a fraction of the earth's land surface can be traversed by wheeled vehicles. Common examples of insurmountable challenges to wheeled motion include rocky mountainsides, rubble fields, disaster sites, and even stairs and ladders. To overcome this limitation, more and more research has been done recently on the possibility of legged robot locomotion. Inspired by the natural capabilities of humans and animals, these legged robots have the potential to traverse a wide variety of terrains, rendering accessible the vast majority of the earth's land surface.

The DARPA Information Processing Technology Office (IPTO) sponsors the Learning Locomotion program, a research effort aiming to develop new and fundamental insights into legged robot locomotion. This program is based around the Boston Dynamics "LittleDog" robot, a twelve degree-of-freedom quadruped robot. Teams at major universities around the country are working on using artificial intelligence techniques to "walk the dog" over extreme terrain using carefully planned and executed footsteps (rather than mindless "flailing"). The sophistication of the required motion makes devising a hand-crafted control scheme for walking over rough terrain prohibitively complex. Instead, artificial intelligence techniques such as machine learning are being used to automatically generate robust and sophisticated control programs, which rival those designed by skilled human engineers.

Learned control programs aim to achieve two objectives: they must generate motions that allow the robot to traverse the terrain in a stable and efficient manner, and they must generate such motions in a timely fashion. Unfortunately, these two goals are often mutually incompatible, because the sophisticated algorithms required to generate the stablest and most efficient routes are characteristically slow and computationally expensive. Quicker algorithms, often using approximate hand-tuned heuristics, can achieve faster speeds but yield less reliable motions, leading to the robot falling over or getting stuck on the terrain. This trade-off between fast control programs and stable motions poses a problem for real-time legged locomotion: how can one achieve reliable robotic walking while operating in real-time?

Objective

I have been working on developing a gait library to facilitate rapid footstep planning. Grounded in the realization that traversing familiar terrain can be done largely by experience without any sophisticated planning, the gait library archives past situations and the planned footsteps for each situation. Once sufficiently many past situations have been stored in the library, machine learning techniques can be used to generate new footstep plans based on the accumulated examples. Once the library is large enough, the new generated plans should in principle be of the same quality as the original plans which it learned from. Furthermore, these new plans can be generated in fractions of a second, simply by querying the library, even if the original plans it learned from required slow and expensive computations. A gait library can offer incredible speeds suitable for real-time (or even faster-than-real-time) applications, while potentially preserving the reliability and stability of more sophisticated and computationally expensive planning algorithms.

Gait Library Design

I designed the gait library prototype in Matlab, for eventual conversion into C++. A separate library is accumulated for each foot, since foot behavior for each foot is qualitatively different. For each recorded footstep, the "context" which is saved includes the local heightmap around the robot, the positions of the four feet, and the desired path, which indicates the intended direction of travel. The context for the i-th training example is stored as a 960-dimensional vector x(i), and the resulting footstep position is stored as a 2-dimensional vector y(i). Archiving the training set into the library involves loading all of the training samples [x(i), y(i)] into a kd-tree, which takes O(m log m) time, where m is the number of training samples. When the planner runs and encounters a new context, and needs to figure out where to step, it queries the kd-tree to retrieve the closest context in the kd-tree to the new context in logarithmic time and returns the footstep position associated with that nearest neighbor. Thus, the gait library essentially operates on a non-parametric 1NN learning algorithm.

The distance between two contexts is calculated using a learned Mahalanobis distance metric. This metric can be optimized to minimize the hold-one-out cross-validation error on the training set; unfortunately, learning an optimal Mahalanobis distance metric for large training sets (even if we only learn the diagonal entries, resulting in a simpler weighted Euclidean distance metric) is time-intensive and (in the general case) computationally intractable. However, once calculated, this learned distance metric can be used to improve the accuracy of the predictions made by the gait library with a negligible per-lookup slowdown.

The initial prototype approach exhibited unacceptably large hold-out and test error, each at roughly 2.9cm average error per footstep. For comparison, the extremely simplistic approach of calculating the mean footstep position and then always taking that footstep yields an error of approximately 3cm -- our approach does only marginally better despite having access to the entire context for the footstep. Clearly, we should be able to do much better than this simplistic approach and its baseline error of 3cm. The unacceptably large hold-out and test errors are indicative of bias in the learning algorithm: we require that the distinguishing characteristics of a given context be linear transformations (or, in the simpler weighted Euclidean case, linear combinations) of its heightmap, current foot positions, and desired trajectory. There is no compelling reason to believe that the distinguishing characteristics of a given context are linear functions of the heightmap, foot positions, and trajectory, and so -- as evidenced by the hold-out error and test error -- this restriction introduces bias into our set of possible hypotheses.

Feature Selection

One strategy to combat this bias is to use better features to represent the footstep context. Using non-linear functions rather than a linear transformation of the heightmap, foot positions, and desired trajectory allows for the feature vector to better capture the essential characteristics of the context, thereby reducing the inherent bias in the learning algorithm. Unfortunately, given the high dimensionality of the input data (960 dimensions or more if larger heightmap patches are incorporated), and the high time cost associated with measuring the quality of particular features (approximately three hours to convert 500 training runs into a gait library and calculate the test error on 500 test runs), running an automatic feature selection algorithm in the space of all possible non-linear functions is not feasible -- instead, good features must be chosen by hand.

Although features must be chosen by hand, they do not have to be chosen randomly. I used two approaches to select potential heightmap features in a principled fashion: first, using vision-inspired features to discern meaningful higher-level topological properties, and second, using PCA to decompose the heightmap into coefficients of eigen-heightmaps. The vision-inspired features captured absolute topological properties of the map (notably edges and gradients) and the PCA decomposition captured the structure of the map in terms of its place in the larger set of observed heightmaps. I also explored several other potential feature functions, such as applying a strict threshold to the heightmap, blurring and de-blurring the heightmap, varying the size of the heightmap patches, calculating the matrix eigenvalues of the heightmap patch, using only the x- or y-coordinates of the desired trajectory, and so on, but these did not yield results as significant as the principled methods above.

Among the vision-inspired features, the most promising were the Canny edges of the heightmap. Using the results of the Canny edge detector as features reduced the test error from ~2.95cm to ~2.45cm on the best foot, and offered an average improvement of ~3.6mm on the four feet. These results were significantly better than those offered by the Sobel, Prewitt, Roberts, or Laplacian of Gaussian edge detectors, although they also took slightly longer per heightmap patch, which resulted in a considerable slowdown for training the gait library on 500 paths (for a total of more than 4000 context/footstep entries per foot). This is not a significant disadvantage: the time to construct the gait library is a one-time cost, and the regime where speed is important -- generating new paths given new contexts -- experiences only a minor slowdown due to the need to generate the Canny edges of the heightmap. Moreover, since the Canny edges of the whole heightmap will not change from step to step, the edge detection cost can be amortized over the entire path, rather than needing to be paid for each step taken.

Performing the PCA decomposition of the training heightmaps was straightforward and computationally tractable. Although memory constraints prevented all of the available heightmaps from being used, enough could be used that the results appeared sound and intuitive (the first principal eigen-heightmap was a flat board, followed by a typical ridge, a typical slope, and a typical valley). Once the eigen-heightmaps have been calculated in this preprocessing step, any new heightmap can quickly be converted into coefficients for the top fifty eigen-heightmaps by taking the scalar projections of the new heightmap onto each eigen-heightmap. Three ways of using these coefficients as features were considered: using them directly, scaling each coefficient by how important its associated eigen-heightmap is, and scaling each coefficient by the square root of its associated importance. The motivation for scaling the coefficients is so that differences in the latter (and less important) eigen-heightmaps are not weighted as heavily as differences in the earlier (and more important) eigen-heightmaps. Scaling coefficients by the square root of the importance is the principled way to do this: when the Euclidean distance between two vectors is taken, if one coefficient has been scaled by the square root of x, the ultimate distance measure will have a weight x on that coefficient. Empirically, the third approach did indeed show the greatest reduction in test error, dropping the test error from ~2.95cm to ~2.6cm on the best foot, and offering an average improvement of ~2.6mm on the four feet.

A Mixed Strategy: Incorporating the Baseline

Additional examination of the error distributions in the initial experiment has indicated that although the mean test error in the prototype approach is comparable to the mean test error in the baseline approach, the mean test error in the prototype approach is much more due to large outliers (in the 12cm - 14cm range) than the baseline approach, which has few test errors above 8cm. For a notable fraction of the test examples, the prototype approach performs extremely well, whereas the baseline approach performs extremely well on relatively few examples. Without even using better features, the prototype approach successfully predicts the footstep position with an error of less than 5mm (essentially perfectly) for 15% of the test samples, while the baseline approach predicts the footstep position with an error of less than 5mm for only 2.5% of the test samples. This indicates that the prototype approach is already considerably better suited toward fine control problems (such as legged locomotion) than the simplistic baseline approach -- which makes sense, because it should be difficult to perform fine control without any context information!

One potential approach toward reducing the number of large outliers in the test error is, rather than competing with the baseline strategy, to incorporate it into the prediction algorithm, along with the gait library. When there is a potential for a large outlier to occur -- for example, when the footstep position predicted by the gait library strongly deviates from the baseline footstep -- we revert back to the baseline prediction. This approach substantially improved the test error; indeed, when combined with Canny+PCA features, it showed exceptional results for those feet whose test errors were not showing much improvement with Canny+PCA features alone. The test errors for feet two and four, which had languished at ~3.27cm and ~3.45cm initially and only improved to ~3.12cm and ~3.28cm with Canny+PCA features, compared with a baseline error of ~3.06cm and ~3.19cm, were improved to ~2.88cm and ~3.05cm respectively using the mixed strategy of Canny+PCA features w/ reverting to baseline on large deviations. The mixed strategy outperformed the baseline on every foot with a margin of at least 1.5mm.

However, the mixed strategy is theoretically unsound. While it may be a good practical heuristic, there is no principled reason to believe that the gait library's predictions are likely to be bad simply because they deviate from the mean step baseline. A better metric might consider the distance between the observed context and the gait library's nearest context, or even use an SVM with a Gaussian kernel to perform multivariate regression on the most important context/footstep support vectors, while discounting outliers and other irrelevant points that happen to get into the training data. This idea is discussed later, as a possibility for future work.

Future Work

Given the results presented thusfar, a natural opportunity for future research on the topic is the continued search for good features for footstep contexts. Even within the space of edge detectors, Canny edges are not the quintessential state of the art -- future work may consider Bergholm, DSC, Iverson, Nalwa, SUSAN, or Rothwell edge detectors (or others). Furthermore, gradient magnitudes and edge detection hardly span the space of modern vision techniques: a comprehensive approach to further research into vision-inspired feature choices should consider corner detectors (such as Harris corners), line detectors (such as Hough transforms) and true sophisticated object-recognition techniques, such as Haar cascades, SIFT, and SURF. One might even consider a parts-based hierarchical object model, representing the heightmap as a set of major hills and valleys which are then encoded as features, or using image segmentation techniques to develop robust topology-inspired features. Vision-inspired features take advantage of the inherent topographic nature of the heightmap data, and I believe they will be crucial to achieving the best possible performance on such data.

One can expand upon the use of PCA as well. Since the PCA decomposition does not rely at all on the elements of the data vectors being topologically related (in the way that image processing techniques assume that one is working with a 2D grid of image points), one can apply PCA not just to the heightmaps but also to the rest of the data in a context: the footstep positions and the desired trajectories. These eigen-contexts will not take advantage of the topographical nature of the heightmap, but on the other hand they may offer the best opportunity for generalizing the results of the gait library to other similar problems: by using PCA and scalar projection rather than intuition-guided trial-and-error to select features, a generalized library system could be built that quickly maps inputs to outputs for arbitrary complex systems given training data. Were one to pursue this line of research, it might be done in conjunction with the use of a regularized multivariate Gaussian-kernel SVM (discussed below). It is unclear to me whether such a general learning/library system could truly be effective across a variety of applications -- but such a system, were it indeed viable, would provide an invaluable speed-boost in many contexts.

The mixed strategy approach was unsatisfying because it relied on a heuristic to determine how to avoid outliers. Rather than using an imperfect mixed strategy, another potential approach toward reducing the number of large outliers in the test error, as mentioned earlier, is switching from a 1NN algorithm to a regularized SVM. A regularized SVM with a Gaussian kernel is essentially a nearest neighbor predictor, with the added advantage that it only selects neighbors from the most important points (the support vectors). A regularization scheme would allow the predictor to drop relatively unlikely outlying points (the ones which might cause large outlying errors) in favor of larger margins. Although most SVMs perform either classification or regression in a single variable, the SVM technique has been extended to work for multivariate regression, and standard packages exist to perform this task (e.g. SVM-Struct, related to the popular SVM-Light). Thus, it may be informative to consider whether the test error on a regularized SVM with a Gaussian kernel is significantly better than the test error on the 1NN library.

Although none of the above approaches were explored in the context of this project due to time limitations and difficulties in converting the existing data structures to the appropriate formats, they represent promising directions for future work on the gait library.