Robot Motion for Obstacle Negotiation

Anish Adukuzhiyil, Harshit Singh, Pavani Vantimitta
Stanford University
{ajohna, harshs, pavani}@stanford.edu

1. Introduction

We seek to test for the capability of a quadruped programmable robot to overcome barriers (generally referred to as the obstacle negotiation problem), given appropriate machine learning algorithm(s) for motion planning and execution. We programmed a robot to run in an environment with different terrains and present results from our trial runs using a combination of Principal Component Analysis (PCA) and supervised learning. Error estimates under varying conditions of the learning algorithms have also been highlighted.

2. Present approaches

In related work, one popular approach has been to apply Reinforcement Learning [1]. Some other approaches have focused on adapting to terrains using mechanically higher designs such as clever mechanical design of legs that automatically adapt to terrains [2,3]. Another different approach has been to use Gait (using a rule-based algorithm); a method of free gait generation for quadrupeds is presented in [4]. There are many more approaches that have been used to tackle this problem.

3. Problem description

Given the task of obstacle negotiation, the challenge is to first identify the obstacles which the robot has to necessarily cross in order to reach its goals, and also develop the set of motions that it will need to execute to successfully overcome those obstacles. In view of this objective, this project involves implementation and evaluation of techniques using which the robot can learn to successfully traverse various kinds of terrains through an iterative process of sensing, planning and execution.

4. Environment Description

The StanfordDog (Figure 1) is a four-legged programmable robot (LittleDog, made available by Boston Dynamics) used for motion planning research at the STAIR lab. The robot comes with a control and planning library with a Matlab interface.

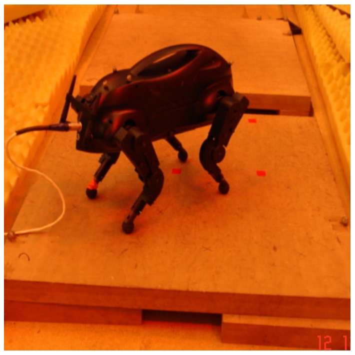
Figure 1: StanfordDog on the Gap terrain

The Vicon Motion Capture (MoCap) system can capture and model the terrain and its environment in a 3D setting. This information is then processed into parameters such as joint angles, body posture and terrain location and made available to the user through the Robot API.

We developed a set of motions keeping in mind the robot's center of gravity and the range of parameter values of each of its leg, hip and knee joints such that no motion execution causes the robot stance to become unstable. All eight motions, along with their defining features, are listed in Table 1.

Note that left and right shuffle are not symmetric

Forward Motion		Backward Motion		Left Shuffle		Right Shuffle		Left Rotation		Right Rotation		Jump Motion (Barrier)		Jump Motion (Gap)	
Distance moved	Angular Shift	Distance moved	Angular Shift	Distance moved	Angular Shift	Distance moved	Angular Shift	Distance moved	Angular Shift	Distance moved	Angular Shift	Maximum Barrier Width	Maximum Height	Maximum Gap Width	Maximum Height
3.97	0.13	4.34	1.48	1.08	1.67	0.95	0.44	0.52	10.6	0.34	9.3	10	11	14	4

Table 1: All distance and height measurements are in centimeters. Theta measurements are in degrees.

even though the parameter values for the atomic maneuvers are chosen to be symmetric. We believe this is to be attributed to differential motor power in the joints. For all our trial runs, we used a fixed step size of 2 for the shuffles, and 1 for all other motions.

An optimal motion, given a current position of the robot, is defined as the motion that the robot should execute in order to bring it closer towards the goal and consequently, closer to any obstacle lying between itself and the goal. When the robot is at the obstacle, the optimal step is then the special maneuver that will let the robot overcome it.

4.1 Heightmaps

The system returns an image representation, which we call 'Heightmap', of the 3-D space of the robot's environment, centered on itself. One way of looking at the Heightmap is as a 2-dimensional image of size N-by-N pixels, where each pixel's index into the matrix is given by the x and y coordinates of the pixel in 3-D space, and the 'intensity' value of each pixel stores the z coordinate of the same spatial representation.

5. Methodology

Identifying terrain features in the locality of the robot is definitely important, but there is an important caveat to be considered. Since the robot is only concerned with getting to the final goal, it should not concern itself with barriers that do not lie in the direct path to this goal. The maximum displacement that a robot can achieve in any motion also limits the neighborhood of interest.

Recognition of the obstacle is not necessary for the purpose at hand. We only need the shape of the obstacle (i.e., its contour in 3 dimensions).

For the purpose of motion, it should not matter if the robot executes a non-optimal motion as long as it does not fall off the terrain board or hits the obstacle. At the edges of the board and near the obstacle, however, it is imperative that the motion executed by the robot is only the one that is deemed to be optimal by the training set.

5.1 Essential Features

Keeping these above in mind, it appears that the set of sufficient features that enable selection of the optimal motion in any position would have to include:

- Representation of objects (possible barriers) in the robot's immediate neighborhood in terms of their position and orientation as concern the robot.
- Direction of the final goal.

The test data should be compared on the basis of the optimized feature space to the learned training data.

5.2 Experimental Setup

For the purpose of our experiment, we used two flat terrain boards, positioned at the origin of MoCap's global coordinates. Between the two boards, we either kept a raised barrier across the breadth of the boards, or maintained a gap between the boards. We use the term 'barrier' to denote a raised bar in the terrain, and the term 'gap' to denote a depressed region within the terrain.

We then collected a set of 300 Heightmaps corresponding to different positions of the robot on the gap terrain. We labeled each of these height-maps with the maneuver determined to be the optimal next motion. These were manually determined, first by looking at the Heightmaps only, and then verified by getting the robot to execute the maneuvers in the given context.

6. Feature Space Reduction

6.1 Initial Pre processing

We processed the Heightmaps in order to reduce the number of irrelevant features being considered.

For a Heightmap, only the z values around an edge were taken as features, while the rest of the pixels were set to 0. We implemented a Gaussian weighting filter to accentuate image features closer to the robot. The processing was carried out in the sequence as shown in Figure 2.

6.2 Feature Extraction

The state-of-the-art in object recognition

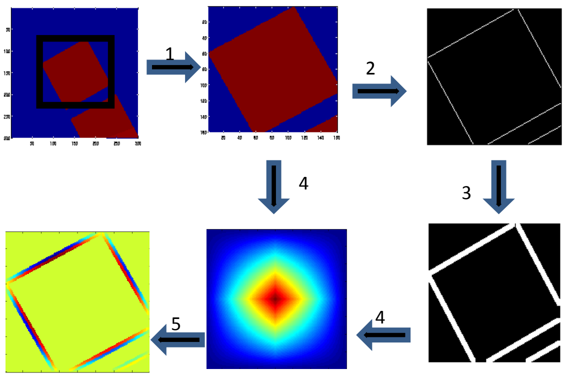

Figure 2: Preprocessing Cycle for the Heightmaps: 1) Slicing 2) Edge detection (Canny) 3) Dilation 4) Element wise multiplication 5) Final feature representation for a Heightmap

algorithms have used decision trees based on feature space transformations such as Haar wavelets [6,7], and SVMs [8]. However, plain image-based comparisons such as these would not work well in our case.

One reason is that these approaches allow for rotational and translational invariance of object shapes, which assumption would lead to wrong results in our case. Also, as argued above, we believe the choice for next optimal motion depends not just on the robot's local environment, but also the direction of the goal and the next barrier, some things that image-based features do not incorporate.

Therefore, we decided to modify a simple image-characteristic based feature representation in order to incorporate the more relevant features. We settled on Principal Component Analysis (PCA) as a good starting point, as it has been used extensively in applications like Face Recognition and Fingerprint Identification.

6.3 Principal Component Analysis

In the base form of this algorithm, we consider each Training Heightmap as a feature matrix, with each pixel position as a unique feature. Across all training examples, the mean and variance of each feature is first calculated and then used to normalize the feature matrix. Now, the eigenvalues and eigenvectors of this matrix are calculated. We then reduce the feature space based on these eigenvalues by thresholding them at some constant.

After thresholding, those eigenvectors whose eigenvalues were not thresholded form the eigen-space of our problem. Now, we take the test Heightmap and find its projection on the eigen-space. The Euclidean distance between this and the each feature vector in the training matrix is calculated. The least of these calculated distances gives the best matching Heightmap from the labeled training set, which we hope would also provide the best maneuver in the current scenario.

Essentially, we are faced with a multi-classification problem, identifying the most appropriate maneuver to be executed on the basis of feature-based similarity of current context to previously seen situations.

6.3.1 Errors in General PCA

As explained, PCA was used to extract only the pertinent features (say k) on a training set by using a threshold on the variance of the features from the mean. These k features then were used as a basis for comparison of a new Heightmap (test data) with the set of training Heightmap data. Thus we find the closest match in the labeled data, and therefore the best optimal maneuver as well.

As shown later, the PCA approach does provide quite accurate results based solely on height-map image characteristics. However, when we used each pixel as a feature, in addition to matching the closest images, it also matched to laterally shifted versions of the input image. This was because the eigenvectors corresponding to an orientation of the robot will be in the same direction. Essentially, different positions of the robot on the terrain can have the same orientation in global coordinates.

6.4 Other Features

We experimented with a direction feature extracted from MoCap's global coordinate system, given a fixed goal (on the other side of the obstacle).

We added another feature – the distance between the barrier and the robot. The first obstacle that comes in the path of the robot should be the most important feature that the robot should consider.

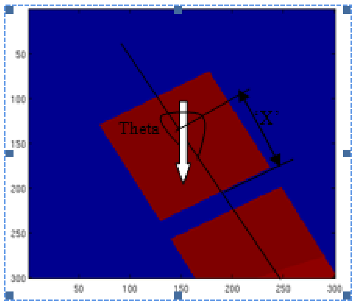

Figure 3.: Distance offset from barrier = x
Orientation to goal = theta

6.5 Pruning

Given a Heightmap in the current context of the robot, we used the value of the current orientation (θ, as labeled in Figure 3) in order to find a subset of the training Heightmaps that also have robot orientation within a certain range of θ.

We later also used the distance between the robot and the first edge of the obstacle between it and the goal (x as labeled in Figure 3) to carry out similar pruning of the Training set.

These pruned sets were then passed onto the PCA module in order to obtain the best match to the current Heightmap.

Figure 4 shows the distribution of Theta and x in our training data set.

7. Optimization

7.1 Initial Selection of constants

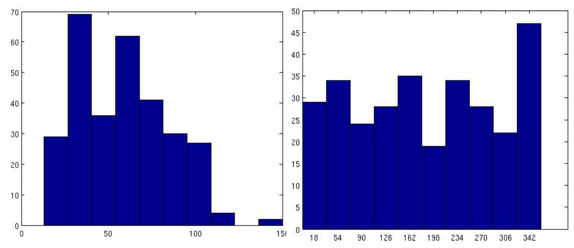
Figure 4: Histograms showing the spread of a) Barrier Distance and b) Orientation angle in the Training Set

The issue of Heightmap size involves a tradeoff between generality and more features. For the images of size 50 by 25 cm that we are using, a threshold value of 0 was needed as the returned Heightmap was most often only a block image composed of very few variation elements. The threshold value can be chosen to best suit different terrains under consideration.

We decided to keep the size offsets of the image from the robots centroid to reflect the robot's view of its surroundings. While this is not being done in absolute terms, we tried to incorporate the possible shifts of view given the robot's set of movements so that it could see atleast as far ahead as the next position it could get to. Since backward/forward motions shift the robot by twice as much as left/right shuffle, we kept the Heightmap aspect ratio at 2:1.

7.2 Sensitivity Analysis

The most important parameters for our algorithm are the Heightmap size on which the PCA is to be performed, the ranges of values around the current value of the robot's orientation (θ), and the range of values around the current value of the robot's distance from the nearest barrier.

From the different test runs, the optimal values of image offset size were found to be 50 and 25 cm, the optimal range of theta was found to be 5 degrees and the optimal range of distance from the barrier was found to be 10cm. The precision recall curve for these values is plotted in Figure 5.

8 Evaluation

There are two types of error measurement techniques that we employed. One is intrinsic evaluation, which we quantified by using precision and recall. The other, and the more pertinent one to the objective of the robot in question, is extrinsic evaluation. This is the fraction of times the robot

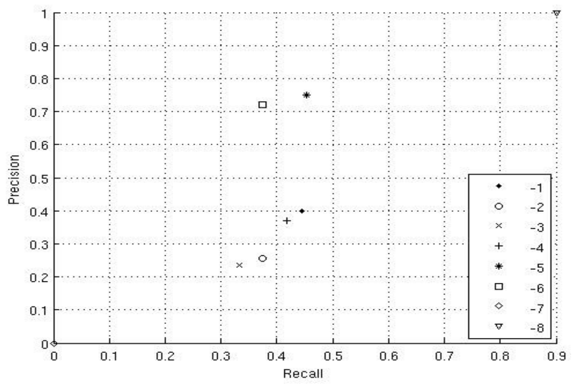

Figure 5: Precision and Recall Curve for image offset of 50, range of theta as 5, and range of barrier distance as 10

successfully crosses the obstacle when starting at a random position on the terrain.

We conducted Leave-one-out validation and obtained precision/recall values to check for accuracy in the predicted optimal motions. We only tested on the gap terrain.

8.1 Intrinsic Evaluation

Using just the precision/recall, we get high values of Fmean for the special jump-gap motion (8 in Fig. 5). This result is good because, as clarified earlier, it is costly for the robot to be wrong as concerns the jump motion.

The precision and recall values for jump=barrier motion (7 in Fig. 5) were 0 as this motion was never labeled as an optimal motion in our training set of gap terrain Heightmaps.

The precision values for left (5 in Fig. 5) and right rotation(6 in Fig. 5) are high (consistent with the desired levels), since they have significant rotation angles which could mean toppling over for positions near the edge.

Since forward (1 in Fig. 5), backward (2 in Fig. 5), left shuffle (3 in Fig. 5) and right shuffle (4 in Fig. 5) motions do not require high precision and recall values because they have been labeled as the optimal next steps for positions of the robot following which it has little danger of toppling off the edges.

8.2 Extrinsic Evaluation

We defined the success rate as the number of times the robot actually overcomes the obstacle. Starting from random initial positions on the terrain, we found the success rate to be 60%.

9 Error Analysis

Features used	Heightmap Slice + Gradient + Gaussian Filtering(100X50)	Heightmap Slice + Gradient + Gaussian Filtering (100X50)	Heightmap Slice + Gradient + Gaussian Filtering (100X50)
Pruning basis used		Orientation Range	Orientation Range + Barrier Distance
No of training examples chosen for PCA	300 (all)	30-40	5-15
Average Fscores for motion classes (1-7)	0.2905	0.3395	0.3540
Fscore for motion class 8 (Jumping the Gap)	0.6957	0.9474	0.9474

Table 4: Comparison of results obtained using different pruning method for PCA selection

We noticed that, often, the robot tended to fall over the edge when its initial position was near the edge, or when a motion brought it near the edge. This means that the sequence of operations to obtain the gradient of the Heightmap were not always useful in this case. Another feature that we need may need is the offset of the robot from the edges of the terrain.

10 Directions for further work

In essence, supervised learning and PCA will work well for familiar terrains. For terrains that have not previously been seen, reinforcement learning would be a good framework to integrate our approach with.

Other work also involves improving upon the maneuver set currently developed, in terms of efficiency and parameters.

Another possible direction is to base the robot's assessment of its surrounding on data/images captured from a sensor/camera placed on the robot itself. This will enable motion planning from the perspective of the robot, and a global motion capture system would not be necessary. We tried to best assess the implications of a robot-centric view by taking as small a Heightmap as possible, but much will depend upon what information the sensor provides.

References

[1] Honglak Lee, Yirong Shen, Chih-Han Yu, Gurjeet Singh and Andrew Y. Ng, "Quadruped Robot Obstacle Negotiation via Reinforcement Learning", in International Conference on Robotics and Automation, Orlando, Florida, USA, 2006

[2] S. Hirose, A. Nagabuko, and R. Toyama, "Machine that can walk and climb on floors, walls, and ceilings," in International Conference on Advanced Robotics, Pisa, Italy, 1991

[3] S. Hirose, K. Yoneda, and H. Tsukagoshi, "Titan vii: Quadruped walking and manipulating robot on a steep slope," in International Conference on Robotics and Automation, Albuquerque, New Mexico, USA, 1997

[4] S. Bai, K. H. Low, G. Seet, and T. Zielinska, "A new free gait generation for quadrupeds based on primary/secondary gait," in International Conference on Robotics and Automation, 1999

[5] Paul Viola, Michael Jones, "Rapid Object DetectionUsing a Boosted Cascade of Simple Features" (2004)

[6] Viola, P. and Jones, M. 2001. Rapid object detection using a boosted cascade of simple features. In Computer Vision and Pattern Recognition, vol. 1, pp. 511–518.

[7] C. Bahlmann, B. Haasdonk, and H. Burkhardt. Online handwriting recognition with support vector machines—a kernel approach. In Proc. of the 8th IWFHR, pages 49–54, 2002.