Stereo Vision and Terrain Modeling for Quadruped Robots

J. Zico Kolter[†], Youngjun Kim[§], and Andrew Y. Ng[†]
[†]Computer Science Department, [§]Aeronautics and Astronomics Department
Stanford University, Stanford, CA 94305
{kolter,youngjun,ang}@cs.stanford.edu

***Abstract*— Legged robots offer the potential to navigate highly challenging terrain, and there has recently been much progress in this area. However, a great deal of this recent work has operated under the assumption that either the robot has complete knowledge of its environment or that its environment is suitably regular so as to be navigated with only minimal perception, an unrealistic assumption in many real-world domains. In this paper we present an integrated perception and control system for a quadruped robot that allows it to perceive and traverse previously unseen, rugged terrain that includes large, irregular obstacles. A key element of the system is a novel terrain modeling algorithm, used for filling in the occluded models resulting from on-board vision systems. We apply our approach to the LittleDog robot, and show that it allows the robot to walk over challenging terrain using only on-board perception.**

I. Introduction

Legged robots offer the potential to navigate challenging, irregular terrain that is inaccessible to wheeled vehicles. Inspired by this potential, in recent years there has been much progress in the field of legged locomotion and quadruped robots in particular. However, in many cases these works have assumed that either 1) the robot has an apriori model of the terrain or that 2) the terrain is simple enough to be overcome using very minimal sensing and/or clever open-loop mechanisms.

Our main contribution in this paper is an integrated perception and control system for a quadruped robot that allows it to perceive and walk over previously unseen, rugged terrain that includes large, irregular obstacles; this is accomplished using only on-board sensors on the robot. While there has been a great deal of past work on vision techniques for legged robots — we discuss these in much greater detail shortly — our system represents a significant step forward in terms of fully autonomous, real-time locomotion on challenging terrains that require careful and deliberate planning of the footsteps. We implement our approach on the "LittleDog" robot, designed and built by Boston Dynamics, and shown in Figure 1. Although there has been a great deal of recent work on this robot for navigating rough terrain [1, 2, 3, 4], nearly all this work has assumed a known model of the environment, and near-perfect state estimation using a motion capture system. Therefore, the methods we present here serve as a means of extending this past work to the more realistic setting where all perception and localization must be achieved using on-board sensors only.

Briefly, our system consists of three components, shown in Figure 2. First, the *perception* component uses stereo vision and a point-cloud matching algorithm known as Iterative Closest Point (ICP) to build and align a single coherent model of the terrain as the robot walks over it, while simultaneously localizing the robot in this model. This builds upon similar past work [5, 6], but applying these techniques to the real-time quadruped vision task is highly non-trivial, and we highlight some of the challenges and solutions specific to our setting. However, even in ideal settings, using this technique alone is insufficient, because the camera cannot see the rear side of obstacles, and hence will always produce an incomplete model of the terrain. But in rocky, unstructured environments, we cannot simply ignore these occlusions, as they may represent pitfalls that the robot should avoid; indeed, all the planning and control methods for the LittleDog referenced above require complete knowledge of the terrain. Therefore, a key element of our approach is the second component of our system, a *terrain*

Fig. 1. The LittleDog robot, designed and built by Boston Dynamics, Inc.

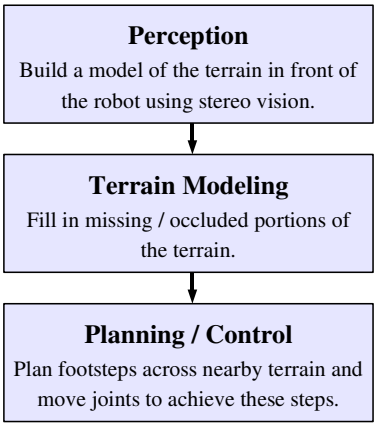

Fig. 2. Overview of the perception and control architecture for the quadruped.

Fig. 3. *(left)* LittleDog robot equipped with a Tyzx DeepSea stereo camera. *(right)* Closer view of the camera.

modeling algorithm that "fills-in" the missing portion of the terrain with a reasonable guess of its structure. Specifically, we propose a novel non-parametric algorithm, based on texture synthesis methods, that allows us to sample from a distribution over the missing terrains portions, subject to the geometric constraints imposed by the camera. Finally, given this model of the terrain the *planning and control* component plans nearby footsteps and executes joint commands so as to achieve these footsteps. This piece of the system is virtually identical to that described in [4] — indeed, we developed the perception and terrain model-ling elements precisely so that these previous techniques could work with minimal modification. However, there are some cases where the perception system necessitated changes in the planning and control, and we highlight these situations below.

The rest of this paper is organized as follows. In Sections II, III, and IV we describe the perception, terrain model-ling, and planning and control components of our system respectively. In Section V we present empirical results, demonstrating the ability of the complete system to cross terrains of varying difficulty; we also demonstrate the benefit of the terrain model-ling algorithm. Finally, in Section VI we review relevant related work, and conclude in Section VII.

II. Perception using Stereo Vision

As mentioned, the goal of the perception component is to build a model of the terrain and localize the robot relative to this model, using an on-board stereo camera. Figure 3 shows the mechanical setup of the stereo camera attached to the LittleDog robot. The system operates as follows: as the robot walks across the terrain, the perception system periodically (in our implementation, twice a second) takes a picture from the stereo camera, and converts this into a mesh of the currently visible portion of the terrain. The system then applies an algorithm called Iterative Closest Point (ICP) in order to align the most recently created mesh with the previous meshes.[1] By repeating this process, we both build a single, coherent model of the terrain and can estimate the position of the robot relative to the terrain (since the camera is rigidly attached to the robot body). Similar approaches have been previously considered in the robotics literature [5, 6], though we are unaware of any previous application of such techniques to quadrupeds in particular. In the remainder of this section we briefly highlight some specific challenges and solutions that came up in the quadruped setting.

In order to assure real-time convergence of the ICP algorithm, we found it was important to align subsequent meshes using a coarse model of the terrain. That is, we uniformly sample from the stereo image at a relatively low resolution and use this sparse point cloud to form the meshes for alignment. However, this coarse model cannot be used for the actual planning and control system, as it has the effect of filling in many the small gaps which can trap the robot if it steps in them. For this reason, we found it critical to maintain *two* separate mesh models of the terrain, one sampled at a low resolution for use in the ICP alignment, and one sampled at a higher resolution that is passed to the terrain modeling and planning/control elements.

In order to prevent the ICP algorithm from getting trapped in a local minima (a common problem for the ICP algorithm), we need to provide it with a good initial guess of the position and orientation of each new mesh. Fortunately, the control system is able to provide a very reasonable initial guess simply by using dead reckoning along the desired trajectory of the robot (which we update whenever the ICP algorithm produces a new estimate of the robot's true state). Furthermore, we found that this initial state estimate is much more reliable if the picture of the terrain was taken at a point when the robot's body was relatively stable. Therefore, in our final version of the system, the controller actually instructs the perception system about when to take a picture. Without initializing the ICP matching in this manner, we found that the resulting meshes would frequently be inaccurate, and useless for localization.

III. Non-parametric Terrain Modeling

When we build a terrain model using this stereo system described above, there will be many missing portions. This is not simply an artifact of the stereo camera being unable to find correspondences in its pair of images, but rather an inevitable effect of occlusions in the camera's line of sight: since the camera is located on the robot, there will necessarily be rear sides of objects that cannot be seen. Figure 4 shows an example terrain as well as the height map formed when using an on-board camera to scan the terrain (the pictures shown here are gathered in simulation, and so represent the presence of holes even with an "ideal" camera). In this section we present a method for filling in the missing portions of the height map, using a non-parametric terrain modeling algorithm.

The method we present here is inspired by texture synthesis approaches from the computer graphics community, in

[1] The ICP algorithm [7, 8] is a method finding a full 6-DOF transformation that will align two overlapping mesh models. Briefly, the ICP algorithm works by first choosing a set of corresponding points on the two meshes, then applying a 6-DOF transform to reduce some error metric, and iterating this process until convergence. For our implementation we used the point-to-plane error metric, given by

$$E = \sum_i \left((Rp_i + T - q_i) \cdot n_i \right)^2$$

where p_i and q_i are corresponding points, n_i is the normal corresponding to p_i, R is a rotation matrix, and T is a translation.

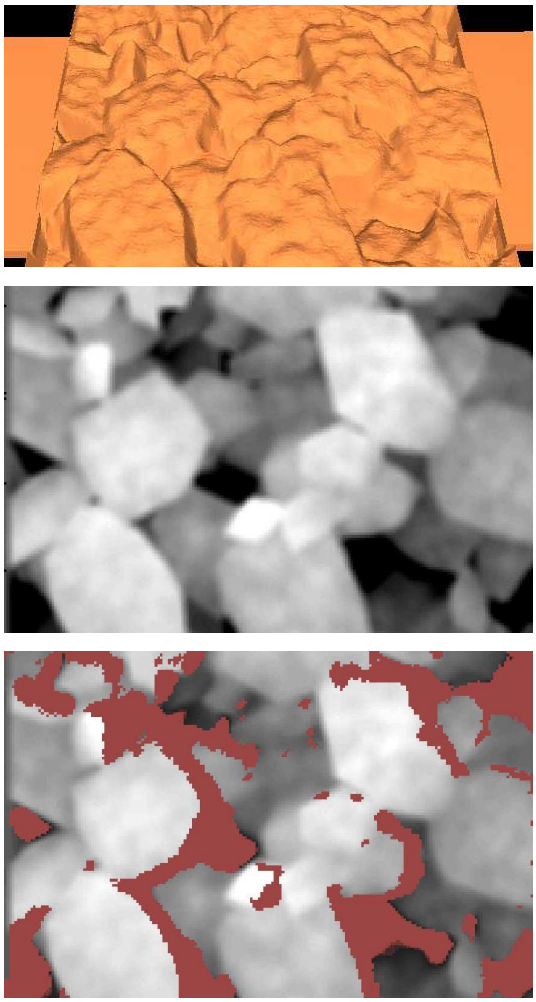

Fig. 4. *(top)* 3D model of sample terrain. *(middle)* Corresponding height map of the terrain, where the color intensity indicates height. *(bottom)* Height map of the model acquired by an on-board camera when crossing the terrain left to right (in simulation); red regions denote missing/occluded areas.

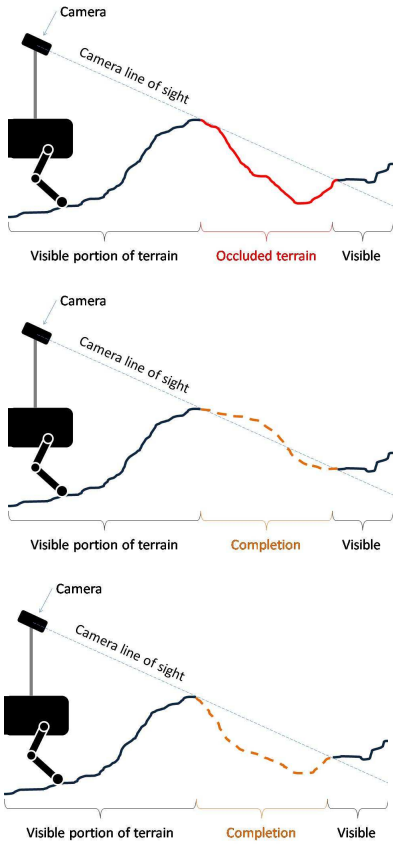

Fig. 5. Visualization of the line of sight constraints imposed by the geometry of the terrain and camera. The middle figure shows a possible completion that violates the line of sight constraint (and would be rejected by step 3 of the algorithm), while the bottom figure shows a completion that obeys the constraint.

particular the method presented in [9], so we begin by briefly describing this method. The algorithm maintains a "library" of known images (in our case, this would correspond to known height maps of terrain). To fill in a missing entries in an image, the algorithm looks in this library to find a patch that closely resembles the visible areas nearby the missing value; it then uses the corresponding pixel from the patch to fill in the missing value. This process is repeated pixel by pixel, until the image is completed. Fundamentally, this can be viewed as an non-parametric method for sampling from the posterior distribution over the missing pixels. But the implicit assumption of this approach is that the unobserved data is missing at random (that is, it assumes that the probability of a value being unobserved does not depend on its value or its neighbors values).

However, in our setting, where the missing values stem from occlusions in the camera's line of sight, there is additional information that we can leverage. Even when a region of the height map is missing, this does not mean we have no information about the height. Rather, we know that all occluded terrain must lie below the camera's line of sight, which connects the last visible region to the next visible region. An illustration of this constraint is shown in Figure 5. What we want, therefore, is to sample from the posterior distribution over the missing region *subject to the geometric constraints imposed by the positions the camera and terrain*. We therefore adapt the non-parametric texture synthesis method discussed above to directly handle such geometric constraints.

A formal description of our algorithm is given in Figure 6, though the basic intuition is quite simple. For each missing x, y value in the height map, we consider a patch of size $w \times w$, centered around that missing value; this is done in step 1 of the algorithm. In step 2, we then draw n random $w \times w$ patches from our library, and compute 1) how much the library patch violates the height constraints and 2) the difference (measured as a weighted sum of squared errors on the visible portions) between the library patch and the height map patch. We add the bias term b here, so that the algorithm only looks at the *relative* heights in the patch, not the global heights. Finally, in step 3 we find the library patch obeying the height constraints that is most similar to the height map patch, and use this to

Algorithm Fill-Terrain $(w, n, W, H, \bar{H}, \mathcal{L})$
Parameters:
$w \in \mathbb{N}$: window size (assumed to be odd)
$n \in \mathbb{N}$: number of samples to draw from library
$W \in \mathbb{R}^{w \times w}$: weighting matrix
$H \in \mathbb{R}^{p \times r}$: height map with missing height values
$\bar{H} \in \mathbb{R}^{p \times r}$: map of maximum heights determined by geometric line-of-sight constraints
$\mathcal{L}$: library of fully known terrain heights
For $x = 1, \ldots, p,\ y = 1, \ldots, r$, **if** H_{xy} missing:
1. Define $A, \bar{A}, B, G, \in \mathbb{R}^{w \times w}$, $m \in \mathbb{R}$
$A_{ij} = H_{x-\frac{w+1}{2}+i, y-\frac{w+1}{2}+j}$
$\bar{A}_{ij} = \bar{H}_{x-\frac{w+1}{2}+i, y-\frac{w+1}{2}+j}$
$B_{ij} = \begin{cases} 1 & A_{ij} \text{ known} \\ 0 & \text{otherwise} \end{cases}$
$m = \sum_{i,j=1}^{w} B_{ij}$
2. **For** $k = 1, \ldots, n$,
- Draw a random patch $C \in \mathbb{R}^{w \times w}$ from library.
- Compute bias term and middle height
$b = \frac{1}{m} \sum_{i,j=1}^{w} B_{ij}(A_{ij} - C_{ij})$
$h_k = C_{\frac{w+1}{2}, \frac{w+1}{2}} + b$
- Compute weighted patch similarity to A
$s_k = \sum_{i,j=1}^{w} W_{ij} B_{ij} (C_{ij} - A_{ij} + b)^2$
- Compute patch constraint violation
$v_k = \sum_{i,j=1}^{w} \max\{B_{ij}(C_{ij} - \bar{A}_{ij} + b)^2, 0\}$
3. Set height using best match, $H_{xy} = h_{k^\star}$, where
$k^\star = \begin{cases} \arg\min_{k \in \mathcal{K}} s_k & |\mathcal{K}| > 0 \\ \arg\min_k v_k & \text{otherwise} \end{cases}$
$\mathcal{K} = \{k : v_k = 0\}$
Return H.

Fig. 6. The Fill-Terrain algorithm for filling in missing portions of a height map subject to geometric constraints

fill in the missing height. This process is repeated until all the heights are filled in.

Figure 7 shows the results of applying this algorithm to the terrain shown in Figure 4, with the regions that were missing in the original image outlined in blue. Notice that when we do not include the geometry constraints, the occluded areas are frequently modeled as flat continuation of the previous terrain. This is due to the fact that in our sample library, flat terrain is more common than sudden drop-offs, so the missing portions are typically estimated to be flat. Of course, this is an unrealistic estimate because if the terrain were flat then we would not have the missing regions to begin with. In contrast, notice that when we include the geometric constraints the terrain drops off in the occluded regions, much like the true terrain. Also notice that these drop-offs are typically more drastic that the drop-offs in the geometric constraints themselves. This is due to the fact that sudden drop-offs are more common in our terrain library than gradual slopes. By employing the non-parametric modeling algorithm, we automatically generate terrain with features similar to that in our training set. We stress the fact that the filled in terrain is *not* contained in the terrain library, but rather the library contains a set of terrains generated by the same random process (described in Section V).

IV. Planning and Control

The aim of the planning and control component is to plan a set of nearby footsteps and move the robot's joint so as to achieve these steps. As we mentioned in the Introduction, this element of the system is virtually identical to the planning and control system we have developed for the LittleDog robot, assuming full knowledge of the terrain [4]. Therefore, we give here only a very high-level description of the system.

From a high level, the planning system first creates a *cost map*, which indicates the desirability of different points on the terrain. Following this, we use receding horizon search to plan a set of nearby low-cost footsteps. Next, we plan trajectories for the robot's COG and feet in order to achieve the footsteps while maintaining static stability of the robot. Finally, we apply PD control using inverse kinematics to move the joints so as to achieve the directories. As mentioned, a much more thorough description is given in [4], but we want to highlight two elements here that are particularly important for understanding how perception integrates with the planning and control.

First, we want to stress the intimate connection between the model of the terrain and the quality of the resulting footstep plans. The cost map for footstep planning is generated directly from the height map, using local features — such as slope, height differential, smoothness — that describe the terrain in patches around each point. Therefore, the cost for a given point on the terrain depends not only on the height at that point, but also on the height of nearby points. Therefore, even if we want to step only in locations seen observed by the perception system, we still need the terrain modeling element of our system, so that we can achieve an accurate estimate of the cost at points that may just border on unobserved regions; indeed, we demonstrate in the next section that using our terrain modeling algorithm significantly improves our estimate of the cost function, even on the observed portions of the terrain.

Second, one significant and unavoidable change that our on-board perception system necessitates is the fact that we can no longer plan complete paths across the entire terrain. Whereas our system in [4] plans a complete set of footsteps from a start to a goal location before ever moving the robot, this is not possible using on-board vision, because we cannot perceive all the terrain until we actually start to move across cross it. Therefore, one significant change in our planning approach for this work is that we plan footsteps for the robot only for the nearby terrain, and continually update this plan as we perceive more and more.

V. Experiments

In this section we present experimental results for our system and algorithms. The chief result of this paper is

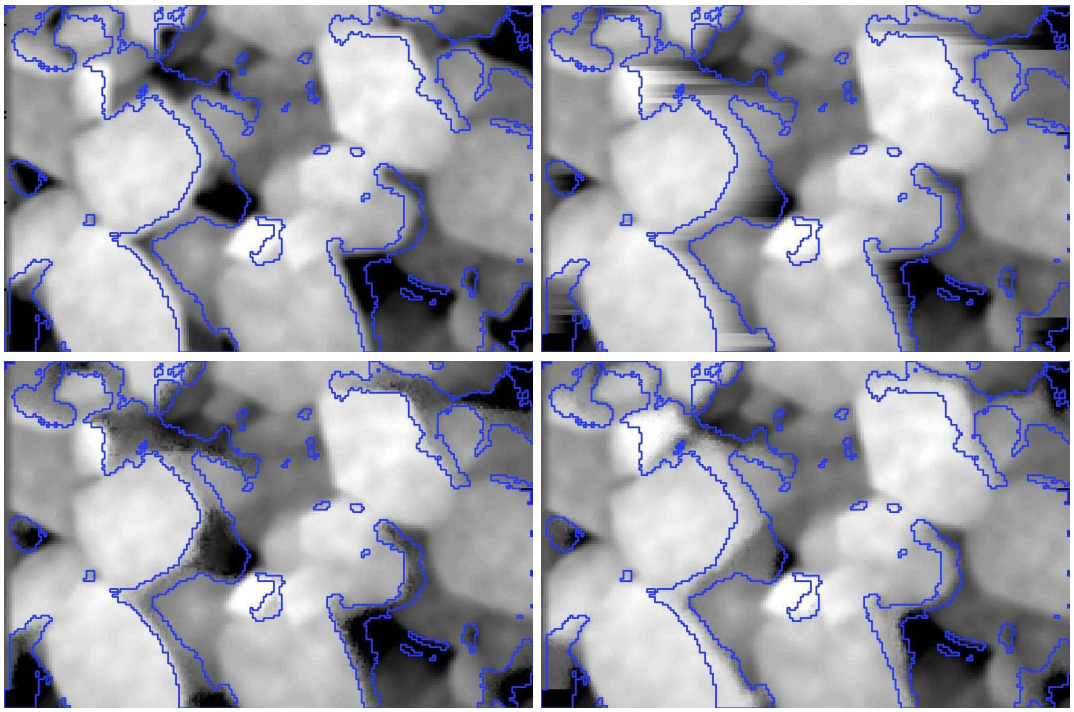

Fig. 7. *(top left)* Height map of the sample terrain. *(top right)* Known upper bound on the terrain height found via geometric constraint. *(bottom left)* Reconstructed height map using our proposed algorithm. *(bottom right)* Reconstructed height map using texture synthesis without geometric constraints. In all cases the blue outline shows the occluded area (see Figure 4) that the algorithms are filling in.

a successful application of these techniques to the task of quadruped locomotion over challenging terrain, using only on-board vision and sensing systems. A video of the robot navigating over this terrain is included with the submission.

A. Evaluation of the Terrain Modeling

Due to the relatively limited amount of real terrain for which we have a complete 3D scan, we perform the primary quantitative evaluation of the terrain modeling algorithm using artificially generated terrain in order to run enough experiments to determine statistical significance. Our experimental setup was as follows: we generated ten 1m x 1m pieces of terrain to for use in the terrain library and 10 more 1m x 1m pieces for evaluation.[2] In a simulated environment, we walked the dog over these evaluation terrains and simulated a perfect range sensor with the same intrinsic and extrinsic parameters as the Tyzx stereo camera. From these ideal range readings we generated a sequence of meshes modeling the terrain, which represents the "ideal" model of the terrain achievable by a range sensor with the the same geometry of the true camera. We created height maps using these meshes, and then applied our terrain modeling algorithm to this height map to fill in any of the missing areas produced by the occlusion of the terrain.

We evaluated the system based on three metrics. By far the most important of these was how well the system could determine costs for visible portions of the terrain. Recall from the previous section that planning footsteps well relies entirely on building a good cost function, which in turn requires a good estimate of the entire terrain, even if we never plan footsteps directly in unobserved areas. Therefore, the most natural way of evaluating the quality of the terrain model-ling algorithm is to see how accurately it recreates the true costs — i.e., the cost map built using the complete model of the terrain. In addition, we also evaluated how well the system predicted costs and heights of the *unseen* portions of the terrain, though these metrics are less important, because in typical situations the we can avoid planning footsteps in these areas.

In addition to our algorithm for terrain modeling, we also evaluated the performance of several other methods. First, for computing the cost at *visible* locations in the terrain, we compared to a method that computes features by simply ignoring the value of any missing heights (for example, one of the terrain features happens to be the standard deviation of heights within a local neighborhood of the point; if a nearby point is unseen, we simply ignore its value for the purposes of computing the standard deviation). Second, we compare the method to what would occur if we treated the geometric constraints on the heights as the heights themselves,

[2]The terrain generation was accomplished as follows. We generated 200 random convex bodies by sampling points from a three dimensional multivariate Gaussian distribution (with random covariance matrix) and finding the convex hull of these points. Using the Open Dynamics Engine (ODE) simulation environment, we dropped these bodies into a 1 meter square area and simulated the environment until the objects came to rest. We then sampled the height of this terrain at a resolution of 1mm, convolved the height map with a random 20x20 matrix to add noise, and scaled the heights to lie in a reasonable range for the quadruped. The end result is terrain that looks very similar to natural rock formations, which is the chief type of terrain that we want to navigate with the LittleDog robot. The terrain depicted in Figures 4 and 7 is an example of a terrain generated in this manner.

Algorithm	RMSE of Cost at Visible Points	RMSE of Cost at Occluded Points	RMSE of Heights at Occluded Points
Our method	**0.1005**	**0.2122**	**0.0201** m
Ignore missing heights	0.1686	—	—
Constraints as Heights	0.1231	0.2148	0.0202 m
Texture synthesis w/o geometric constraints	0.1407	0.2380	0.0264 m
Optimal constant value	0.2472	0.2261	0.0364 m

TABLE I

ROOT MEAN SQUARED ERROR BETWEEN THE TRUE HEIGHT OR COST AS MEASURED BY FULL HEIGHT MAP AND THE PREDICTION OF THE DIFFERENT METHODS DESCRIBED IN THE MAIN TEXT. ALL RESULTS ARE AVERAGED OVER 10 TESTING TERRAINS.

and used their values to compute the features. Finally, we also compared to the texture synthesis method *without* the geometric constraints and to the best constant value for the unknown costs and heights.

Table I shows the results of our algorithm and the others evaluated on the criteria described above. In all cases we compute costs or heights and then compare these to the true costs and/or heights generated by the full height map and computed the root mean squared error between our prediction and the true value. On all the metrics, our proposed method outperforms the other methods. On the most important metric, the error between the estimated cost and the true cost at visible points on the terrain, our algorithm outperforms the next best method (using constraints as heights), with a p-value of $p = 9.2 \times 10^{-5}$ using a pairwise t-test. Furthermore, in a qualitative sense, as exemplified in Figure 7, the predictions output by the our approach typically look much more natural than any of the other methods.

B. Evaluation of the Complete System

In this section we present a successful application of the methods presented here on the LittleDog quadruped robot. In particular, we use the system presented here to successfully navigate over two terrains of varying difficulty, without any prior models of the terrain or use of the motion capture system. This result is also shown in the video accompanying our submission.

The two terrains used for evaluation are shown in Figure 8. The first terrain is a milled model of several flat rocks stacked together, with a maximum height differential up of about 6cm. The second terrain is a model of several less structured, more jagged rocks, with a maximum height differential of about 12cm; successful navigation over this terrain requires extremely accurate placements of the feet. To the best of our knowledge, crossing this harder terrain with the LittleDog robot *requires* perception of some kind; despite numerous efforts we have been unable to cross this terrain without careful planning of the footsteps — without such planning the feet of the robot are virtually guaranteed to slip into a crack, causing the robot to fall. Indeed, through many discussions with other researchers working on the LittleDog, we feel that this terrain is difficult to cross even *with* perception. Therefore, crossing this terrain using only on-board vision represents a significant achievement for the LittleDog.

Figure 9 shows several snapshots of the robot autonomously

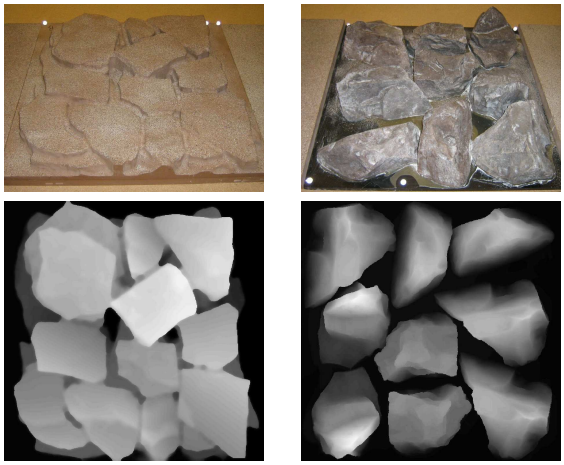

Fig. 8. Pictures and height maps of the two terrains on which we demonstrate the system.

crossing these two terrains using the system described in this paper. Figure 10 shows the model of the terrain generated by the system as well as the estimated path of the robot over the terrain. The RMSE between the mesh generated by the stereo vision and the scanned model of the terrain is only 1.809mm. Discontinuities in the path correspond to locations where the dead reckoning estimate is updated by the ICP position. As can be seen, although the position estimate almost always need to be corrected somewhat, the error is typically not excessive, implying that the state estimate produced by dead reckoning in between the ICP updates remains relatively accurate. Finally, although the preceding section provides a more quantitative evaluation of the terrain modeling algorithm, the benefit is apparent on the real system as well. After numerous trials, the system will invariably fail if we do not use the terrain modeling component of the system.

VI. RELATED WORK

There has been a vast amount of previous work on legged locomotion and quadruped locomotion in particular, so we provide here only a brief overview. We have already discussed much related work on the LittleDog robot [2, 1, 4]. There has also been work by Chitta et. al in proprioceptive localization for the LittleDog [10], where the robot must localize itself using only internal sensors, but this differs from the our own work in that they still assume a known model of the terrain. Lastly, there has also been work on running the LittleDog in

Fig. 9. Snapshots of the robots crossing the two terrains used for evaluation.

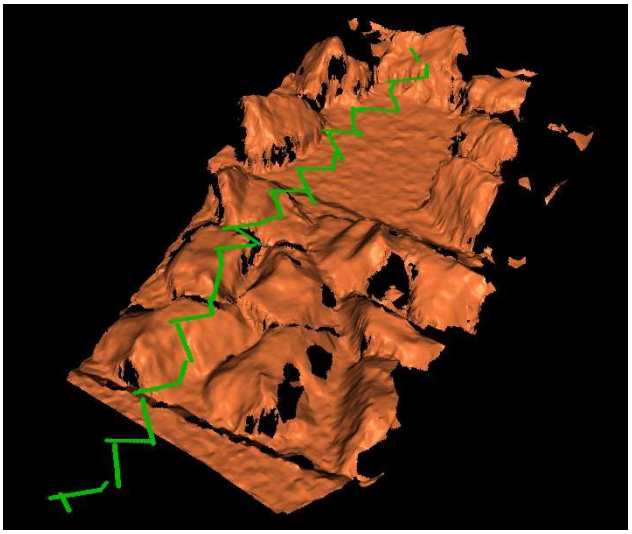

Fig. 10. Model of the terrain and estimated position of the robot as it crosses the harder of the two rock terrains.

unknown environments using on-board vision [11]. However, this work is quite different from what we propose here, as it focuses just on monocular vision, where the goal is to navigate across fairly flat, regular terrain, with colored markings rather than the rugged, highly irregular terrain we consider here.

Other research in quadruped locomotion has focused on using "central pattern generators" (or CPGs), which use networks of biologically inspired neural controllers to achieve periodic locomotion [12, 13, 14]. Some of these robots have been able to walk outdoors in unstructured environments, but the size of the irregularities is typically very small. In contrast, in this work we are concerned with autonomous navigation over highly challenging terrain, with steps equal to the size of the robot's clearance from the ground. There has also much work on dynamic gaits for quadrupeds, both for large robots [15, 16] and small [17, 18]. While such motion is typically much faster than the static walking gait we consider in this work, it is also much less stable, making a static gait more suitable for the challenging terrain we consider. Finally, there are also many "bio-inspired" robots such as RHex [19], RiSE [20] and many others, which are capable of navigating extremely challenging terrain, but do so through the use of clever mechanical design rather than the precise external perception that we focus on here.

Lastly, while much of the current research on legged locomotion has largely ignored the issue of visual perception, there a few notable exceptions in this area. One of the first such robots was the Ambler robot [21], which used a laser range-finder, a simple point-matching algorithm, and dead reckoning to build a map of the surrounding terrain. The Dante II robot [22] also made use of a laser range scanner to map its surrounding terrain, and used this for planning or for teleoperation if the robot determined that the terrain was too challenging. However, our work differs substantially from these projects in many areas: we focus on real-time creation of terrain models using stereo vision as the robot walks, we use a much smaller robot platform, and we explicitly deal with the problem of occluded terrain from an on-board sensor.

Finally, there has been work in the graphics, vision, and robotics communities on texture synthesis, back-face filling, terrain modeling, and other related topics. As mentioned previously, our method for modeling the terrain is based on the texture synthesis algorithm in [9]. There has also been recent work in robotics on terrain modeling [23] in particular. Like our method, this work uses a non-parametric model to represent the terrain, though they do so by using a Gaussian Process, where the key challenge is representing both the smooth and discontinuous properties of standard terrain. However, they do not consider the line-of-sight constraints that are central to our approach.

VII. Conclusion

In this paper we presented an integrated perception and control system for a quadruped robot, based upon on-board stereo vision. One particularly important part of the system is a terrain modeling algorithm, that fills in occluded areas of the terrain. This algorithm is specifically intended for the partially occluded models that arise due to the line-of-sight constraints of on-board robot sensors, such stereo cameras or laser scanners. We apply our method to the LittleDog robot, and show that it enables the LittleDog to walk over

challenging, rugged terrain. This represents a significant step forward in terms of on-board vision for legged robots that require careful planning of the footsteps.

REFERENCES

[1] D. Pongas, M. Mistry, and S. Schaal, "A robust quadruped walking gait for traversing rough terrain," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2007.

[2] J. R. Rebula, P. D. Neuhaus, B. V. Bonnlander, M. J. Johnson, and J. E. Pratt, "A controller for the littledog quadruped walking on rough terrain," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2007.

[3] M. Stolle, H. Tappenier, J. Chestnutt, and C. G. Atkeson, "Transfer of policies based on trajectory libraries," in *Proceedings of the International Conference on Intelligent Robots and Systems*, 2007.

[4] J. Z. Kolter, M. P. Rodger, and A. Y. Ng, "A complete control architecture for quadruped locomotion over irregular terrain," in *Proceedings of the IEEE International Conference on Robotics and Automation (to appear)*, 2008.

[5] J. Diebel, K. Reutersward, S. Thrun, J. Davis, and R. Gupta, "Simultaneous localization and mapping with active stereo vision," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2004.

[6] L.-F. Gao, Y.-X. Gai, and S. Fu, "Simultaneous localization and mapping for autonomous mobile robots using binocular stereo vision system," in *International Conference on Mechatronics and Automation*, 2007.

[7] Y. Chen and G. Médioni, "Object modeling by registration of multiple range images," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 1991.

[8] S. Rusinkiewicz and M. Levoy, "Efficient variants of the ICP algorithm," in *Proceedings of the Third International Conference on 3-D Digital Imaging and Modeling*, 2001.

[9] A. A. Efros and T. K. Leung, "Texture synthesis by non-parametric sampling," in *Proceedings of the IEEE International Conference on Computer Vision*, 1999.

[10] S. Chitta, P. Vernaza, and D. Lee, "Proprioceptive localization for a quadrupedal robot on known terain," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2007.

[11] M. N. Dille, "Free littledog!: Toward completely untethered operation of the quadruped robot," Carnegie Mellon, Tech. Rep. CMU-CS-07-148, 2007.

[12] Y. Fukuoka, H. Kimura, and A. H. Cohen, "Adaptive dynamic walking of a quadruped robot on irregular terrain based on biological concepts," *The International Journal of Robotics Research*, vol. 22, pp. 187–202, 2003.

[13] S. Peng, C. P. Lam, and G. R. Cole, "A biologically inspired four legged walking robot," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2003.

[14] H. Kimura, Y. Fukuoka, and A. H. Cohen, "Adaptive dynamic walking of a quadruped robot on natural ground based on biological concepts," *The International Journal of Robotics Research*, vol. 26, no. 5, pp. 475–490, 2007.

[15] M. H. Raibert, *Legged Robots that Balance*. MIT Press, 1986.

[16] J. G. Nichol, S. P. Singh, K. J. Waldron, L. R. P. III, and D. E. Orin, "System design of a quadrupedal galloping machine," *International Journal of Robotics Research*, vol. 23, no. 10–11, pp. 1013–1027, 2004.

[17] B. Hengst, D. Ibbotson, S. B. Pham, and C. Sammut, "Omnidirectional locomotion for quadruped robots," in *RoboCup 2001: Robot Soccer World Cup V*, 2002, pp. 368–373.

[18] N. Kohl and P. Stone, "Machine learning for fast quadrupedal locomotion," in *Proceedings of AAAI*, 2004, pp. 611–616.

[19] U. Saranli, M. Buehler, and D. Koditschek, "Rhex: A simple and highly mobile hexapod robot," *Int. Journal of Robotics Research*, vol. 20, pp. 616–631, 2001.

[20] A. Saunders, D. Goldman, R. Full, and M. Buehler, "The RiSE climbing robot: Body and leg design," *Proc. SPIE Int. Soc. Opt. Eng.*, vol. 6230, p. 623017, 2006.

[21] E. Krotkov and R. Simmons, "Perception, planning, and control for autonomous walking with the ambler planetary rover," *Intl. Journal of Robotics Research*, vol. 15, no. 2, pp. 155–180, April 1996.

[22] J. Bares and D. Wettergreen, "Dante II: Technical description, results and lessons learned," *International Journal of Robotics Research*, vol. 18, no. 7, pp. 621–649, July 1999.

[23] T. Lang, C. Plagemann, and W. Burgard, "Adaptive non-stationary kernel regression for terrain modeling," in *Proceedings of Robotics: Science and Systems*, Atlanta, GA, USA, June 2007.