Feature-Based Terrain Classification For LittleDog

Paul Filitchkin and Katie Byl

***Abstract*— Recent work in terrain classification has relied largely on 3D sensing methods and color based classification. We present an approach that works with a single, compact camera and maintains high classification rates that are robust to changes in illumination. Terrain is classified using a bag of visual words (BOVW) created from speeded up robust features (SURF) with a support vector machine (SVM) classifier. We present several novel techniques to augment this approach. A gradient descent inspired algorithm is used to adjust the SURF Hessian threshold to reach a nominal feature density. A sliding window technique is also used to classify mixed terrain images with high resolution. We demonstrate that our approach is suitable for small legged robots by performing real-time terrain classification on LittleDog. The classifier is used to select between predetermined gaits to traverse terrain of varying difficulty. Results indicate that real-time classification in-the-loop is faster than using a single all-terrain gait.**

I. INTRODUCTION

Terrain classification is a vital component of autonomous outdoor navigation and serves as the test bed for state-of-the-art computer vision and machine learning algorithms. This area of research has gained much popularity from the DARPA Grand Challenge [1] as well as the Mars Exploration Rovers [2].

Recent terrain classification and navigation research has focused on using a combination of 3D sensors and visual data [1] as well as stereo cameras [3] [4]. [5] uses vibration data from onboard sensors for classifying terrain. Most of this work has been applied to wheeled robots, and other test platforms have included a tracked vehicle [6] and a hexapod robot [3]. On the computer vision spectrum of research, interest in terrain classification has been around since 1976 [7] for categorizing satellite imagery. More recent work has focused on the generalized problem of recognizing texture. Terrain and texture classification falls into the following categories: spectral-based [8] [9], color-based [1] [10], and feature-based [11].

Over the last decade, a large volume of work has been published on scale invariant feature recognition and classification. Scale invariant features have proven to be very repeatable in images of objects with varying lighting, viewing angle, and size. They are robust to noise and provide very distinctive descriptors for identification. They are suitable for both specific object recognition [12] [13] as well as broad categorization [14] [15].

To the best of the authors' knowledge, the proposed SURF BOVW recognition approach has not been previously applied to terrain classification. This work aims to provide a terrain classification framework that is more robust than color-based classification and is suitable for realtime applications. In addition, we propose a novel sliding window-based technique for classifying mixed terrain images with high resolution.

II. ALGORITHMS

1) Feature Extraction: In this work we use SURF features for classification. The process of extracting features starts by detecting key points at unique locations on the image. Areas of high contrast such as T-junctions, corners, and blobs are selected and then the neighborhood around each point are used to compute the descriptor. SURF relies on a fast-Hessian detector and selects a region around each key point to compute the descriptor. The descriptor is created using the Haar wavelet response in the horizontal and vertical direction. More details on this process are available in [16].

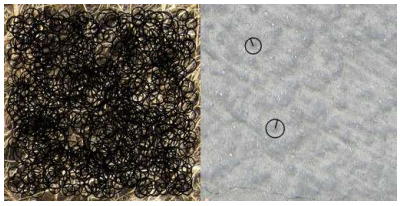

Fig. 1. High Feature Count Variance Using a Constant Hessian Threshold

A pitfall of using a constant Hessian threshold for detection, especially for images of varying frequency content, is the large variance in the number of key points. Figure 1 shows SURF key points for two images with the same parameters. A threshold that is too high may lead to a BOVW with very little data and threshold that is too low will flood the classifier with redundant information and noise. In order to combat this problem we propose a gradient descent inspired threshold adjuster. Let $n = d(h)$ where $d(\cdot)$ is an unknown function that returns the number of key points, n, for a given Hessian threshold h. To find the threshold that produces a desired key point count, h', an iterative convex optimization is performed. For a given step i the error can be written as $e(i) = d(h_i) - d(h')$ and our goal is to minimize this quantity. We use an update method similar to gradient descent, but instead of computing the function's gradient we use the error value divided by the local rate of changes, $\frac{e(i)}{|h_i - h_{i-1}|}$. Equation 1 is the threshold update step where α is a user set value that determines the update rate.

$$h_{i+1} = h_i + \alpha \frac{d(h_i) - d(h')}{|h_i - h_{i-1}|} \tag{1}$$

In practice this approach has the potential to overshoot the target key point count, $d(h')$, by changing the threshold too rapidly. When an overshoot condition is detected, the new threshold becomes the average of the previous two and the update rate is halved.

A. Visual Vocabulary

To generate a vocabulary, we use the k-means clustering algorithm with the initialization procedure outlined in [17]. In the context of this work, the k-means problem is to find an integer number of centers that best describe groupings of descriptor data. More formally: let $k \in \mathbb{Z}$ be the desired number of clusters and let $X \subset \mathbb{R}^M$ be the set of descriptors. The goal is to find k centers (descriptors) in $C \subset \mathbb{R}^M$ that minimize ϕ (Equation 2).

$$\phi = \sum_{\mathbf{x} \in X} \min_{\mathbf{c} \in C} ||\mathbf{x} - \mathbf{c}||^2 \tag{2}$$

The iterative k-means algorithm (Lloyd's algorithm) for achieving this operates as follows: given centers C assign each descriptor $\mathbf{x} \in$

P. Filitchkin and K. Byl are with the UCSB Robotics Lab (http://robotics.ece.ucsb.edu), ECE Dept., 4150 Harold Frank Hall, Santa Barbara, CA 93106 USA e-mail: filitchp@gmail.com and katiebyl@gmail.com

X to the $\mathbf{c} \in C$ that has the smallest Euclidean distance. Then each center is repeatedly recomputed until the centers stop shifting. This procedure will always terminate, but in practice it is common to set a maximum allowable number of iterations. In this work we use the k-means++ algorithm which uses a probabilistic initialization procedure followed by Lloyd's algorithm. In k-means++ the initial center $\mathbf{c}_1 \in C$ is chosen uniformly at random from X and subsequent points $\mathbf{c}_i$ are chosen with the probability in Equation 3. Where $D(\mathbf{x})$ is the distance between $\mathbf{x}$ and the nearest center $\mathbf{c}_j$ that has already been chosen $(0 < j < i)$.

$$P(\mathbf{c}_i = \mathbf{x}' \in X) = \frac{D(\mathbf{x}')^2}{\sum_{\mathbf{x} \in X} D(\mathbf{x})^2} \tag{3}$$

This approach tends to select centers that are further away from the ones that have already been chosen. The idea is to keep a high variance within the centers, but maintain some probability of not always choosing the furthest point.

B. Homogeneous Classification

Once a visual vocabulary has been created each image can be described by a word frequency vector. First, a vocabulary word $\mathbf{v}_i$ is assigned to each descriptor $\mathbf{d}_j$ in the image by choosing the i that minimizes $||\mathbf{v}_i - \mathbf{d}_j||$. This process essentially approximates each descriptor with the vocabulary word that has the nearest Euclidean distance. Each word is then counted and the frequency of each word is stored in a vector $\mathbf{q} \in \mathbb{Z}^n$ where n is the number of visual words. Images in the training set each have a corresponding frequency vector and are used to train the linear SVM classifier.

The training goal of a linear SVM is to find a hyperplane that provides the maximum margin of separation between classes. Let the training set consist of k points $(\mathbf{q}_i, y_i)$ indexed by i where y_i determines if the word frequency vector belongs to the given class. If it belongs to the class then $y_i = 1$ otherwise $y_i = -1$. Any hyperplane that separates the data can be described by all vectors, $\mathbf{h}$, that satisfy $\mathbf{w} \cdot \mathbf{h} - b = 0$ where $\mathbf{w}$ is the vector normal to the hyperplane and b is a scalar bias. The solution for the optimal hyperplane is achieved through quadratic programming using the constraints in Equation 4.

$$\begin{aligned} &\min_{\mathbf{w},\mathbf{b},\xi} \left\{ \frac{1}{2}||\mathbf{w}||^2 - p\sum_{i=1}^{k} \xi_i \right\} \\ \text{subject to} \quad &y_i(\mathbf{w}^T\mathbf{q}_i + b) \geq 1 - \xi_i \\ &\xi_i \geq 0 \end{aligned} \tag{4}$$

Where $p > 0$ is the penalty parameter of the error term. A more complete and generalized introduction to SVMs is provided in [18]. Once a hyperplane is computed for each class, the intersections are used to describe the decision boundaries for the classifier.

C. Heterogeneous Classification

In order to classify a heterogeneous terrain image, it is necessary to generate features with an approximately constant density across all pixels. This is achieved by dividing the image into squares and applying feature extraction on each square independently. The feature sets for each square are then combined to form a single feature set. Afterwards, points on the image are selected on a constant grid and classification is performed in each neighboring region. At each point we iteratively resize a ball so that it encompasses the target number of features (within some tolerance). This procedure is very similar to the gradient descent inspired algorithm mentioned in Section II-.1. Let r_i represent the radius of the ball at the ith iteration and let $D(r_i)$ represent the number of features exclusively inside the ball. The term $D(r')$ is the desired number of features where r' is the unknown radius of interest. To find r' we use Equation 5 which includes a user selectable update rate, α.

$$r_{i+1} = r_i + \alpha \frac{D(r_i) - D(r')}{|r_i - r_{i-1}|} \tag{5}$$

Once a suitable number of features is encircled, a word histogram vector is generated, and classification is performed using the linear SVM classifier trained on homogeneous terrain images. All pixels exclusively in the circle are labeled with the classification result and this step is repeated about each point. Afterwards a voting procedure is applied to each pixel by tallying the number of votes for each class. This procedure is outlined in pseudo code in Algorithm 1.

TABLE I
HETEROGENEOUS CLASSIFICATION DEFINITIONS

Symbol	Description
I	Input image, $I \in \mathbb{R}^{m \times n}$
C	Set of classification centers (pixels) where $C \subset \mathbb{R}^2$
r, r_o	Radius in (fractional) pixels, $r, r_o \in \mathbb{R}$
F	The set of all features for an image I
$F_{\mathbf{c},r}$	All features around $\mathbf{c} \in C$ with radius $\|\|\mathbf{c}\|\| < r$
L	The set of all class labels in the dataset
V_ℓ	Matrix of votes for label $\ell \in L$, $V_\ell \in \mathbb{Z}^{m \times n}$
R	Matrix with classification results, $R \in \mathbb{Z}^{m \times n}$
s_a	Grid spacing for feature extraction in pixels
s_b	Grid spacing for classification points in pixels
$\mathbf{n}$	Desired range (min/max) of features within radius, $\mathbf{n} \in \mathbb{Z}^2$

Algorithm 1 Heterogeneous Terrain Classification

```
r ← r_o
F ← ExtractGridFeatures(I, s_a)
C ← GenerateClassificationPoints(I, s_b)
for all c ∈ C do
    (F_{c,r}, r) ← FindRegionalFeatures(C, F, n, r)
    ℓ ← Classify(F_cr)
    V_ℓ ← Vote(V_ℓ, c, r)
end for
for all ℓ ∈ L do
    R ← CountVotes(V_ℓ, R)
end for
```

III. OFFLINE EXPERIMENTS

A large set of outdoor terrain images was used to benchmark our classification algorithms. A consumer 10 megapixel camera was used to take pictures of six different terrain types: asphalt, grass, gravel, mud, soil, and woodchips (Fig. 2). Images were selected to make this dataset challenging to classify with simply color or texture alone. For instance, the asphalt class includes snapshots of pavement and sidewalk with white, yellow, and red painted lines. Many images also contain small patches of other terrain. Soil pictures include outcroppings of rocks or weeds, and woodchip images include long pine needles resembling blades of grass. To create a suitably large quantity of entries for testing, each raw camera image was divided into sub-images. Seven datasets were created for testing using square images ranging from 192 pixels to 576 pixels in width at 64 pixel increments. In each case, one third of the images were used for training; the rest were used for testing.

Experiment	Dataset	Parameter	Range	Incr.
Vocabulary	384 pixel	Number of vocabulary words	10-400	10
K-means	384 pixel	Number of k-mean iterations	5-200	5
Features	320 pixel	Number of nominal features per image	20-300	10
Size	-	Image size (square dimensions)	192-576	64

TABLE II
LIST OF OFFLINE EXPERIMENTS

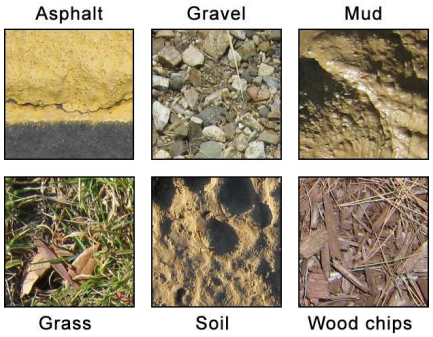

Fig. 2. Examples of homogeneous terrain

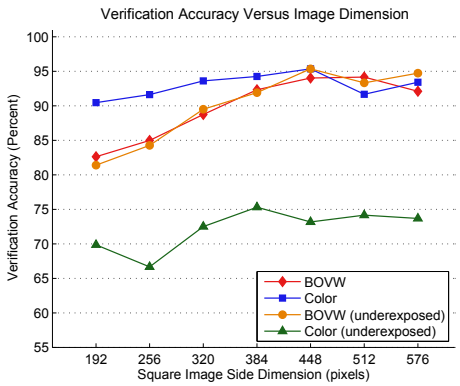

Fig. 3. Size experiment verification and time performance

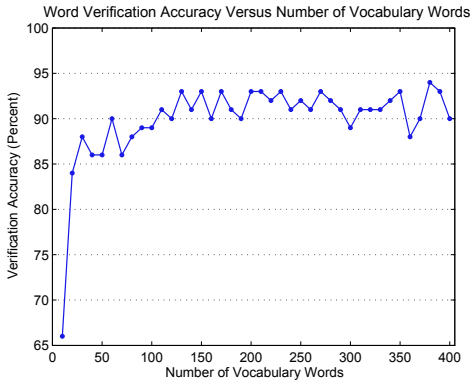

Fig. 4. Word experiment verification results

A. Methodology

The terrain classification framework presents a large number of tunable parameters that impact the performance of feature extraction, vocabulary creation, and classification. This section includes a detailed overview of experiments performed for improving the verification accuracy. Table II lists experiments that were conducted by varying a single parameter while fixing all others. In order to provide a fair comparison between image sizes, the nominal feature density was kept at a constant 1 feature per 640 square pixels in the size experiment. Two sets of verification images were used with the image size experiment. The first consists of unaltered test images selected at random and the second contains identical images that are underexposed via post-processing. This was done to test the robustness of each classification method to changes in lighting. In addition to the above experiments, we compared three aforementioned approaches for generating features: fixed threshold, contrast stretching, and an adaptive threshold adjuster.

B. Results

Intuition suggests that very small images will not have sufficient data for properly classifying terrain in the image size experiment. This is clear when visually inspecting images smaller than 192x192 pixels because it becomes difficult, even for a human, to classify them. As image size gets large the classifier will likely train to a very specific set of visual word histograms and we expected to see the verification error increase. Results matched our expectations and we observed that the optimal image dimension seems to be 448 x 448 pixels (Figure 3). The data also indicates that the color classifier is much less effective in underexposed lighting conditions. This is intuitive because a color histogram simply tallies the color intensity for each pixel and varying illumination alters the color distribution.

We found the number of visual words in the vocabulary to have a direct correlation with the verification accuracy and that very few words would not allow for the classifier to accurately represent each terrain type. Conversely, we found a point of diminishing return as the number of words increased. Accuracy generally improved with word count up to about 150 words as demonstrated by Fig. 4.

We expected the verification accuracy to increase with K-means iterations. Lloyd's algorithm will converge in a finite iterations, so we can assume that after a certain point additional iterations will not be beneficial. There has been a lot of work emphasizing the importance of establishing a good visual vocabulary for recognition [19] [12]; however, in this application, the number of clustering iterations had little effect on verification accuracy.This is a surprising result and may be due to the broad diversity of descriptors compared to the relatively small number of classes. It may also be that the probabilistic initialization procedure in the k-means++ algorithm is a very effective method for this classification problem. We did not notice a point of diminishing return since the cluster centers did not converge during testing.

Similar to the other parameters, we predicted that there needs to be an adequate number of features per image. On the other extreme, too many features per image will introduce noise and redundancy into the system diminishing any further benefits. The results followed our predictions as shown in Figure 5, and we found that for the 320 pixel dataset between 200-250 features per image

produced the best trade-off between size and performance.

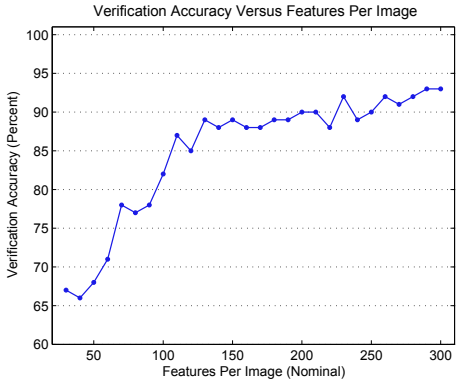

Fig. 5. Feature experiment verification results

The dynamic threshold adjuster proved to be a valuable algorithm because it not only improved verification accuracy, but also greatly decreased memory requirements for the database (Table IV). Verification accuracy, compared to a fixed Hessian threshold, improved by 6 percent while memory requirements decreased by 63 percent. We note that maintaining a nominal number of words per image ensures that the classifier is trained with consistent data and that noise and redundancy is kept to a minimum.

Method	Verification	Size
Fixed Threshold	89.4%	68.1 Mb
Contrast Stretching	90.7%	53.1 Mb
Dynamic Adjuster	95.4%	25.5 Mb

TABLE III
FEATURE EXTRACTION PERFORMANCE

While there is no straight forward performance metric for heterogeneous terrain classification, our algorithm generated visually intuitive results (Figure 6). The algorithm consistently identified homogeneous patches for all classes except for woodchips and gravel. In some cases, boundaries led to misclassification, evident in the bottom row of images. Overall, the algorithm classified images on a fine resolution and generated promising results suitable for future path-planning work.

IV. REAL-TIME EXPERIMENTS

Boston Dynamics' LittleDog quadruped robot (figure 7) was used as the real-time test platform for homogeneous terrain classification. The robot was fitted with a high definition USB webcam with auto-focus capabilities, the Logitech C910. A mid-range laptop was dedicated to terrain classification, and an additional laptop was used to run high-level planning and gait generation. Two machines were used during experimentation to make testing more convenient (dedicated processing, additional screen resolution, etc.), but the framework easily transitions to a single machine.

A. High-level planning

This system is coordinated by a simple Matlab script that schedules terrain classification and selects the gait regime. The script communicates with the other processes using TC/IP sockets found in the Instrument Control Toolbox.

LittleDog starts off each classification cycle in the current gait (in the first cycle this is simply a halted pose) and pauses to

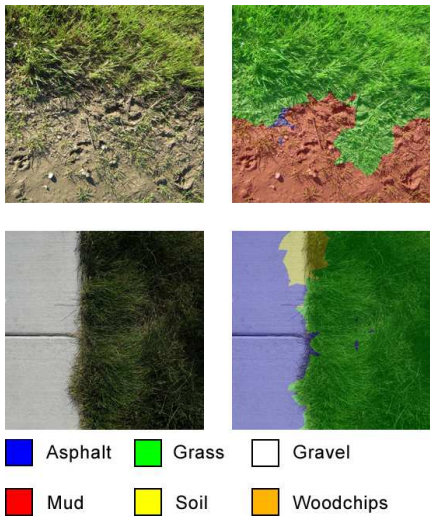

Fig. 6. Heterogeneous Classification Results

Fig. 7. Boston Dynamics' LittleDog

initiate terrain classification. The pause is necessary because we encountered image motion blurring problems with our camera due to low shutter speed. An adequately long pause is required for the camera to stop shaking and the pause is held after initiating classification to ensure system latency does not cause a picture to be taken during movement. After pausing the current gait is continued because the robot has not reached the new terrain. Once it reaches the terrain that has been classified, LittleDog executes the new gait.

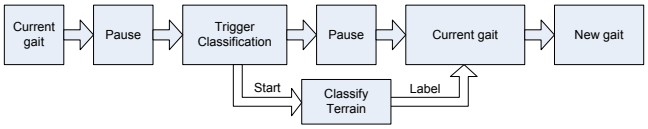

Fig. 8. Real-time execution cycle

B. Terrain Classification

The terrain classification process (figure 9) uses an event-driven framework that includes a graphical user interface (GUI) front-end. The GUI primarily acts as a user monitoring tool for viewing the image input as well as the status of the terrain classification process. A fixed interval "snapshot" timer is then used to sample camera images such that the current image is stored in memory and displayed to the GUI. When a terrain classification event is triggered the latest image is selected for homogeneous classification. The remainder of

the process uses the methods outlined in the algorithms section. Thread support, socket communication and most GUI elements were implemented using the wxWidgets software library. All other image processing and display functionality was implemented using the OpenCV software library.

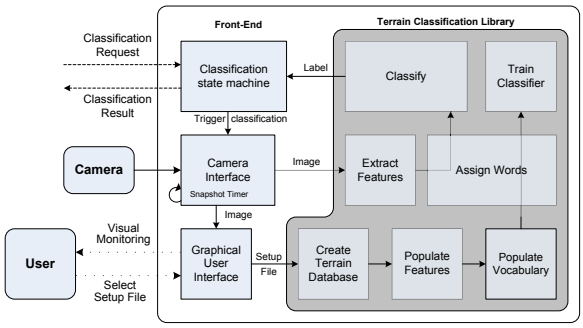

Fig. 9. Real-time terrain classification process

C. Gait Generation

Gait generation was handled by an independent process which selects from several pre-generated gait patterns. The fundamental motion of each leg consists of seven key coordinates in Euclidean three-space: two stance points and five swing points. Each point is also paired with a desired time and then interpolated by a spline. The LittleDog interface library then computes the inverse kinematics for each interpolated coordinate and moves the legs to the desired location. Gait A is reserved for the default Boston Dynamics parameters, however this gait does not demonstrate any advantageous characteristics, and it is only suitable for a nearly flat walking surface. Consequently, it is not used during testing. Gait D is optimized for speed on relatively flat surfaces, gait C is designed for high clearance on rough terrain at the expense of speed, and Gait B is a mixture of the two.

D. Test Environment

In order to create a controlled test environment several 2-by-2 foot terrain boards/containers were created to mimic five natural terrain types. Figure 10 shows close-up images of each terrain type taken using the Logitech C910 webcam. The types of terrain were chosen to provide sufficient visual diversity and offer increasingly difficult walking surfaces. The rubber tile terrain (figure 10A) vaguely resembles a sidewalk surface and provides the least walking difficultly because it has excellent traction and an even surface. The grass-like surface (B) is also mildly challenging and provides a distinctive texture and color. The small rocks in figure 10C are glued to the 2-by-2 boards in order to eliminate slipping, yet they still provide a jagged walking surface. The rubber chips are made from painted recycled car tires resembling a woodchip surface. Lastly, the large rocks, commonly known as Mexican beach pebbles, (Figure 10E) present the greatest challenge because are uneven and shift around very easily. Three 2-by-2 foot terrain boards were each created for the rubber tile, grass, and small rocks. In order to test out a challenging loose terrain transition, the rubber chips and large rocks were put into the same six-by-two foot container.

Initial tests consisted of timing the robot on each type of terrain using each of the three gaits. This established gait performance on every surface and became the basis of gait selection for classification in-the-loop. After collecting time results the visual terrain classifier was tested on the same four courses. The final course

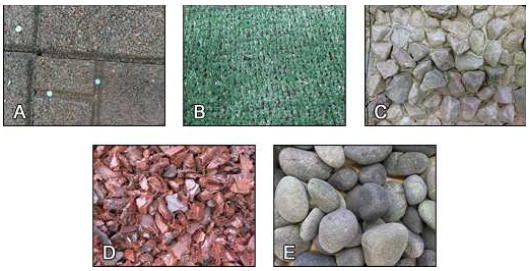

Fig. 10. LittleDog Experiment Terrain

included four terrain types in a 10-by-2 foot area and was tested with real-time classification. This terrain course was used solely for testing the robustness of the visual classifier. To test visual terrain classification each 2-by-2 foot board was subdivided into a 3-by-3 grid and 640x480 pixel pictures were take of each square. The camera was positioned such that the width of the frame captured the width of each square and it was ensured that no overlapping pixels showed up in any image. Images for both the test and training set were chosen at random with two thirds of the images going to the training set. In addition, a down-sampled set of 320x240 pixel images was created to see how well classification would work with less resolution.

E. Results

Visually, each terrain is very distinct and offers a great deal of texture so we did not foresee any problems during offline testing. Offline test results confirmed that our experimental terrain was very well suited for classification and we were able get 100% verification accuracy on both the 640x480 and 320x240 image sets.

		Terrain			
		Small Rocks	**Rocks/Chips**	**Grass**	**Tile**
Gait	B	22.0	-	20.8	20.7
	C	27.7	29.3	26.4	25.8
	D	17.2	-	15.8	17.1
	Classify	22.9	44.2	20.4	22.1

TABLE IV
LITTLEDOG REAL-TIME TERRAIN TRAVERSAL TIMES MEASURED IN SECONDS

Gait D was designed to have very low ground clearance and a relatively fast gait period, so naturally we expected it to perform well on grass and rubber tile. We predicted that LittleDog would not be able to traverse the other terrains because it would stub its feet against the protruding surfaces and veer off course. This was not the case as illustrated by Table IV-E. To the contrary gait D performed the best on all terrain except for big rocks and rubber chips. On the other extreme, our expectation was that Gait C would work reasonably well for every terrain at the expense of speed. This was confirmed by our results, and the only test that took more time was the classification trial for big rocks / rubber chips. We also predicted that Gait B, the hybrid gait, would perform the best on the small rock terrain due to its medium clearance and speed. Our results determined that the hybrid gait was actually unnecessary since Gait D performed faster on the small rock terrain.

Figure 11 shows a time comparison between Gait C and the classification trial. Results indicate that classification was helpful in all cases except for the most difficult terrain. Classification time was slower in that case due to the stopping requirement. If this

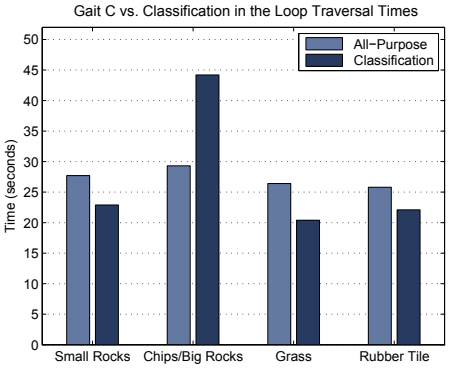

Fig. 11. LittleDog terrain traversal performance

requirement did not exist, we would expect to see a nearly identical classification time as Gait C.

Fig. 12. Real-time classification results

During real time testing the camera auto-focus did not immediately react to changes in object distance. LittleDog would, on occasion, tip the camera downward during a stance and cause the picture to go slightly out of focus. Image blurring generally decreases the performance of the fast Hessian feature detector so we expected to see some misclassifications. Figure 12 shows a consecutive classification trial that was performed on four types of terrain. Tests showed that the classifier was in fact robust enough to minor image blurring (as demonstrated by image 3), and in general there was a very low incidence of misclassifications.

V. CONCLUSIONS AND FUTURE WORKS

In this work we developed novel algorithms that assist popular feature-based recognition techniques. Feature extraction was improved with our gradient descent inspired detection algorithm. Our algorithm increased classification accuracy by 6 percent and decreased memory requirements by 63 percent. A sliding window classifier was also created to identify patches of heterogeneous terrain. This technique showed promising offline results, and set a foundation for future path planning work for legged robots.

The results in this work demonstrated the effectiveness of our feature-based terrain classification framework through offline and real-time testing. Offline experiments provided valuable data on classification performance in a controlled environment. The findings allowed us to select parameters that gave the most desirable mixture of accuracy and performance. Real-time testing showed that our methods effectively aide autonomous navigation on the LittleDog platform. The robot was able to traverse terrain faster with classification in-the-loop despite requiring stops to prevent camera blurring. The only case where this slowed the robot was on the most difficult terrain which already required the slowest gait. Overall, we demonstrated that our homogeneous classification approach works with a wide range of terrain images and can be effectively used in real-time to aide quadruped navigation. The classification results in this work do not have any measure of confidence and we plan to use the support vectors in the classifier to produce such a metric. In the heterogeneous case the voting step can also be used to assign a measure of consensus within areas of overlapping classification. It would also be interesting to try out different classification methods such as boosting and artificial neural networks with the BOVW framework. In addition, a hybrid classifier can be used to combine color with features.

VI. ACKNOWLEDGMENTS

This work is supported in part by DARPA (Contract No. W911NF-11-1-0077). We would also like to thank Boston Dynamics for providing continuing support for LittleDog.

REFERENCES

[1] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, and et. al., "The robot that won the DARPA Grand Challenge," *J. Field Robotics*, vol. 23, pp. 661–692, 2006.

[2] Y. Cheng, M. Maimone, and L. Matthies, "Visual odometry on the mars exploration rovers - a tool to ensure accurate driving and science imaging," *IEEE Robotics Automation Mag.*, vol. 13, no. 2, pp. 54 –62, 2006.

[3] A. Chilian and H. Hirschmuller, "Stereo camera based navigation of mobile robots on rough terrain," *Intell. Robots and Systems (IROS)*, pp. 4571–4576, 2009.

[4] R. Manduchi, A. Castano, A. Talukder, and L. Matthies, "Obstacle detection and terrain classification for autonomous off-road navigation," *Autonomous Robots*, vol. 18, pp. 81–102, 2004.

[5] C. Brooks and K. Iagnemma, "Vibration-based terrain classification for planetary exploration rovers," *Robotics, IEEE Transactions on*, vol. 21, no. 6, pp. 1185 – 1191, 2005.

[6] M. Luetzeler and S. Baten, "Road recognition for a tracked vehicle," in *SPIE Enhanced and Synthetic Vision*, vol. 4023, no. 1. SPIE, 2000, pp. 171–180.

[7] J. S. Weszka, C. R. Dyer, and A. Rosenfeld, "A comparative study of texture measures for terrain classification," *IEEE Trans on Sys., Man and Cyber.*, vol. SMC-6, no. 4, pp. 269 –285, 1976.

[8] X. Liu and D. Wang, "Texture classification using spectral histograms," *IEEE Trans. Image Proc.*, vol. 12, pp. 661–669, 2003.

[9] N. Sebe, M. S. Lew, and N. Bohrweg, "Wavelet based texture classification," in *Int. Conf. on Pattern Rec.*, 2000, pp. 959–962.

[10] J. D. Crisman and C. E. Thorpe, "Color vision for road following," in *Vision and Navigation: The CMU Navlab, C. Thorpe (Ed.* Kluwer Academic Publishers, 1988, pp. 9–24.

[11] A. Angelova, L. Matthies, D. Helmick, and P. Perona, "Fast terrain classification using variable-length representation for autonomous navigation," in *Comp. Vision and Pattern Rec. (CVPR)*, 2007, pp. 1–8.

[12] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in *Comp. Vision and Pattern Rec.*, 2006, pp. 2161–2168.

[13] S. Gammeter, L. Bossard, T. Quack, and L. V. Gool, "I know what you did last summer: object-level auto-annotation of holiday snaps," *Int. Conf. on Comp. Vision*, 2009.

[14] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray, "Visual categorization with bags of keypoints," in *Workshop on Statistical Learning in Computer Vision, ECCV*, 2004, pp. 1–22.

[15] T. Deselaers, L. Pimenidis, and H. Ney, "Bag-of-visual-words models for adult image classification and filtering," in *Int. Conf. Pattern Rec. (ICPR)*, 2008, pp. 1–4.

[16] H. Bay, T. Tuytelaars, and L. V. Gool, "SURF: Speeded up robust features," in *9th European Conf. on Computer Vision*, 2006.

[17] D. Arthur and S. Vassilvitskii, "K-means++: the advantages of careful seeding," in *ACM/SIAM Symp. on Discr. Alg.*, 2007, pp. 1027–1035.

[18] C.-W. Hsu, C.-C. Chang, and C.-J. Lin, "A practical guide to support vector classification," http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf, 2003. [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf

[19] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Object retrieval with large vocabularies and fast spatial matching," in *Comp. Vision and Pattern Rec. (CVPR)*, 2007, pp. 1–8.