

COMPUTATIONALLY INEXPENSIVE APPEARANCE BASED TERRAIN LEARNING IN UNKNOWN ENVIRONMENTS

Prabhakar Mishra¹ and Anirudh Viswanathan²

¹*PES Centre for Intelligent Systems, PES Institute of Technology, BSK Stage - III, Bangalore
Karnataka, India
email: prabhakar.mishra@pes.edu*

²*Robotics Institute, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh,
PA 15213, USA,
email: anirudh@cmu.edu*

Abstract

This paper describes a computationally inexpensive approach to learning and identification of maneuverable terrain to aid autonomous navigation. We adopt a monocular vision based framework, using a single consumer grade camera as the primary sensor, and model the terrain as a Mixture of Gaussians. Self-supervised learning is used to identify navigable terrain in the perception space. Training data is obtained using pre-filtered pixels, which correspond to near-range traversable terrain. The scheme allows for on-line, and in-motion update of the terrain model. The pipeline architecture used in the proposed algorithm is made amenable to real-time implementation by restricting computations to bit-shifts and accumulate operations. Color based clustering using dominant terrain texture is then performed in perception sub-space. Model initialization and update follows at the coarse scale of an octave image pyramid, and is back projected onto the original fine scale. We present results of terrain learning, tested in heterogeneous environments, including urban road, suburban parks, and indoors. Our scheme provides orders of magnitude improvement in time complexity, when compared to existing approaches reported in literature.

1 Introduction

This paper describes a Computer Vision algorithm developed to identify navigable terrain, for autonomous rover operation in unknown environments. Autonomous rovers deployed on remote planetary surfaces, to function as surrogate explorers need to be endowed with the ability of in-motion mapping and localization. This capability is central to the mission objectives which decides the system configuration and payload. The utility of such intelligent systems lie in minimal query and decision support from an Earth based mission operator.

For successful exploration of the environment,

the perception module should be able to distinguish between regions of free-space, from those corresponding to obstacles. Additionally, practical concerns impact the choice of the sensor suite based on size, power, and operability constraints ([21]). The on-board scientific instrumentation, specific to the mission, are allocated significant pay-load volume, weight, and power resources. The perception subsystem, in turn, is constrained for minimization of operability, power budget, and on-board computational resources. This motivates our case to investigate methodologies for optimizing the perception module. In this paper, we restrict the sensor suite a single consumer grade camera, with a proximity

sensor for near-range obstacle detection. We restrict our investigation to enhancing monocular vision based algorithms. Use of multiple sensors and probabilistic multi-sensor data fusion though necessary in applications of this complexity, is not included in the scope of this work.

Terrain characterization, typical for navigation applications, is associated with the relief profile, rock abundance, and soil granularity [10]. In the monocular framework, these engineering parameters are associated with models in the perceptual space, such as terrain hue and color. A limited Earth based solution is to use a-prior terrain models, trained using examples of off-line data, an approach adopted by [8]. However, such classifiers when restricted to a finite number of learned models, become error prone with illumination and terrain surface variations over the mission lifetime.

An on-line terrain classification scheme in perception space is proposed by [7]. This system is similar to our perception architecture, and learns color models of navigable terrain using self supervised learning. The significant difference between the two, is at the training stage. The work by [7] relies on using a laser scanner to identify navigable regions in perception space. Our proposed scheme, is restricted to monocular vision, and emphasizes computationally inexpensive methods, that are orders of magnitude faster than previous approaches.



Figure 1. Typical terrain types in we expect to identify regions of free-space for autonomous rover navigation. Dynamic urban roads (a), unstructured outdoor (b,c), and cluttered indoor environments (d). (a) is from the standard RAT-SLAM dataset, (b-c) are custom datasets, and (d) is a test case from the IDOL dataset.

To overcome the limitations of algorithms proposed in literature, and achieve implementation amenable to real-time resource constraints, we propose schemes that adapt to in-motion learning of terrain types. Figure 1 shows typical terrain in different environments. Our approach relies on a self-supervised learning algorithm, trained by pre-filtering pixels in the perception space. The pre-filtered pixels are assumed to correspond to regions of free-space immediately in front of the rover. The vision algorithm learns dominant colors, as a Mixture of Gaussians. We propose sub-space clustering, implemented using bit-shifts, to learn color models of maneuverable regions. Additionally, a horizon detection scheme restricts the model update related computation to the ground region. The combined strategies of learning models in scale space, with look-up tables to establish the pixel-wise mapping between scales, allows for real-time throughput which is orders of magnitude faster as compared to the technique given in [7].

2 Previous Work

Previous work in learning models to represent terrain, are a part of the Defense Advanced Research Projects Agency (DARPA) programs. Other programs include a terrain classification module targeted at driver assistance systems and autonomous vehicles.

The reader is directed towards the survey paper by [9] for early work in vision based mobile robot navigation. The authors describe perception schemes successful terrain learning schemes, applicable in both structured and unstructured environments.

More recently, three DARPA programs, namely the Grand Challenge (DGC - 2004), the Urban Challenge (DUC - 2007), and Learning Applied to Ground Vehicles (LAGR - 2004–2008) were involved with autonomous mobile robot navigation ([13]).

The DGC involved autonomous driving across 132 miles of desert terrain, the results of which are summarized by [5]. The winner of the DGC was Stanford University's team. The architecture describing Stanley, the team's entry is described by [20]. The vision module ([7]), enabled far-range

terrain classification using probabilistic data-fusion across a laser range finder, and monocular vision. Training data is delineated by the laser scanner, to identify traversable terrain in front of the robot. Subsequently, pixels corresponding to the training region are used to model terrain. The solution allows the vehicle to identify far-range navigable terrain, while driving at high speed.

The DUC involved autonomous navigation in urban environments. The teams had to complete a circuit spanning 96 km, while obeying traffic rules, avoiding other competitors, and moving traffic (Editorial, J. Field Robotics[2]). Additionally, vehicles had to accurately localize the curb, relative to the road, to avoid rollover. The typical solution used by teams included stereo-vision, GPS based localization, and laser scanners to successfully maneuver the urban environment. Carnegie Mellon University Boss, won the DUC, and the robot architecture is described by ([22]). The perception system employs a sensor layer, and a fusion layer. An object hypothesis set is used in dynamic obstacle detection and tracking, and is updated using the sensor layer. The fusion layer implements global classification and object management, for successful navigation on urban roads.

LAGR ([12], [13]), led teams into terrain learning in outdoor, unstructured environments. The LAGR platform was equipped with stereo-vision, proximity-sensors, inertial measurement unit, and infrared rangefinders. The nature of forested terrain and dense foliage led to the development of techniques for in-motion terrain learning ([23]). The technique employs three stages : feature learning, feature training, and terrain prediction. Feature learning is performed by identification of points which correspond to the ground plane, using LIDAR data. The training stage uses a visual classifier to label obstacles, and regions of free-space. This is followed by the predictive stage, which allows for high speed terrain recognition. [11] employs labels from the stereo-camera, which are used to train a deep hierarchical network, and predict real-time traversability. [3] describe methods to short-range and long-range terrain classification. The long-range terrain classification is image-based and learns geometry-based terrain appearance.

All of the above solutions, emphasize use of multiple sensors, and multi sensor data-fusion tech-

niques. Consequently, are not directly applicable to our monocular vision framework.

2.1 Contributions

The key-contributions of this work lie in:

- Monocular vision based terrain learning.
- Speedup of upto 60x in pixel labeling, processing real-time HD video.
- Computationally inexpensive terrain model and update.
- Pipeline based perception architecture - sequential individual stages and pixel level parallelization.
- Computations restricted to integer-only arithmetic, and bit-shift based clustering.

3 Algorithm Description

The monocular vision perception scheme is a pipeline based design. The individual stages in the pipeline are amenable to parallelization, for operations at the pixel level, as shown in Figure 2. The 5 major parts of the pipeline are as follows:

- A Formation of octave pyramid.
- B Pre-filtering near-range pixels, which correspond to navigable terrain.
- C Color based K-Means pixel clustering, at the coarse scale version of the image.
- D Horizon detection – removal of sky region.
- E Scoring pixels in perception space, by learned models.

We describe each stage of the pipeline individually.

3.1 Formation of octave pyramid

An image pyramid as shown in figure 3, is formed for each frame f , in the incoming video feed. Such a pyramid representation, denoted by g_k , at level k is helpful, since the search space for color based clustering is smaller at the coarser level of the



Figure 2. An overview of the algorithm – with individual stages and their corresponding outputs. The processing pipeline is sequential and each stage enables parallel processing at the pixel level.

pyramid. Decimation is performed for each image, to reduce the resolution. We formulate the down-sampling scheme as described by [18]. The original image, f , is convolved with a low-pass filter, h (to avoid aliasing), evaluated at every r^{th} sample. Pixel coordinates are denoted by the ordered-pair (i, j) . The decimation process is implemented using (1).

$$g(i, j) = \sum_{k, l} f(k, l) h(i - rk, j - rl) \quad (1)$$

Where the smoothing kernel $h(k, l)$ is the binomial filter, introduced by [6]. We use the binomial kernel as in 2.

$$h = \frac{1}{16} [1|4|6|4|1] \quad (2)$$

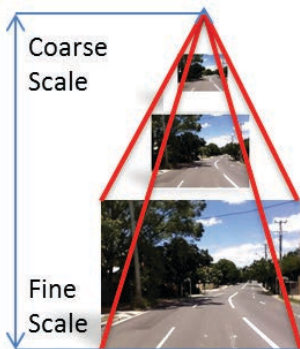


Figure 3. A short section of DNA: two nucleotide aggregates

The convolution operation can be implemented using bit-shifts and addition operations, and is computationally inexpensive. This is illustrated in fig-

ure 4. In our specific implementation, we choose 5 levels for the pyramid. At the coarse scale, the image is $\frac{1}{32}$ times the original size. Color based clustering is now performed on the coarse scale of the octave pyramid.

3.2 Pre-filtering near-range pixels

This section is based on our previous work in [15]. Our perception scheme assumes the following properties of the environment:

- Obstacles are different in color, from the dominant terrain shade.
- The immediate vicinity, of 50cm, in front of the rover is navigable.

The justification for the premise in our argument, lies in the fact that we operate a proximity sensor, for near-range obstacle detection. The proximity sensor module is functionally independent of the perception module, and triggers the control mechanism to execute a turn-in-place maneuver, till a clear field of view is obtained. The discussion of near-field obstacle avoidance control schemes is beyond the scope of this paper.

A computationally inexpensive pre-filter is used to identify pixels at the coarse level, which correspond to navigable terrain. The details of our previous approach [15] is summarized below.

The pre-filter stage is implemented using Region of Interest (ROI) based processing. The ROI operator is a rectangular window X , of dimension

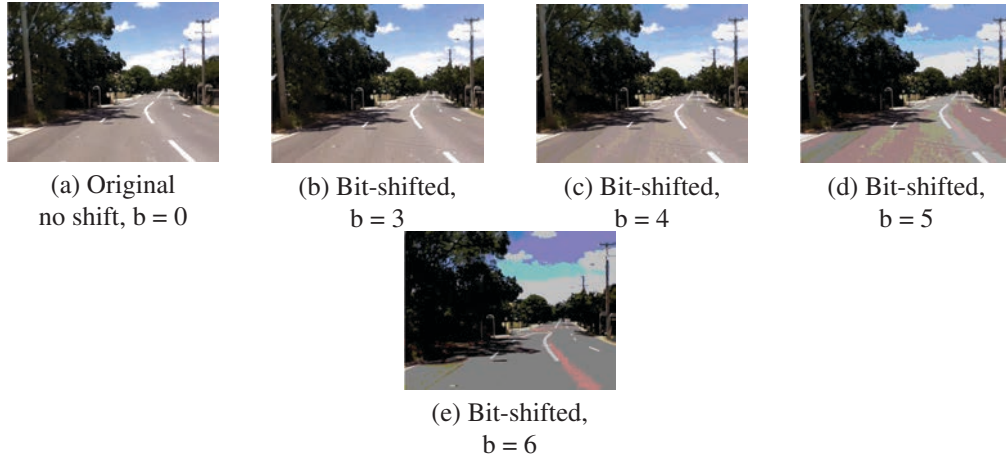


Figure 4. Bit-shifted perception sub-space. The original and bit-shifted versions of the image are shown in (a)-(e). Perceptible quality degradation, and false contouring occurs after 5 shifts. Consequently we choose $b=4$, a trade-off between computation throughput and perceptual image quality.

$2^p \times 2^q$, as shown in figure 5. In our specific implementation, we choose the window to be a size 64×128 pixels, at the fine scale.

Formally, let $\{(u_1, v_1), (u_2, v_2), \dots, (u_m, v_m)\} \in S$ represent the position of pixel inliers in the ROI. The i^{th} pixel in color space, is represented by $g(u_i, v_i) \in \mathbb{R}^3$. Consequently, pixels in the ROI are represented by $X \subset \{S \cap f\}$. Individual pixels, pre-filtered, in the ROI, are represented by $x(u, v) \in X$.



Figure 5. Pre-filtering based on the Region of Interest (ROI), highlighted in orange. The image on the right are pixels which fall within one standard deviation of ROI mean.

Additionally, we adopt the following convention in our discussion:

- μ_X – Mean of the ROI.
- $P(x_{u,v})$ – Image histogram, at the pixel intensity value $x_{u,v}$.

The mean pixel value in the active window is given by (3):

$$\text{Mean: } \mu_X = \sum_{(u,v) \in S} z_{u,v} P(z_{u,v}) \quad (3)$$

To reduce the computational complexity, the ROI is a window of size 64×128 pixels. Consequently the above equation reduces to an accumulate step in (4), and a bit shift in (5).

$$\text{Accumulate: } \mu_X = \sum_{(u,v) \in S} z_{u,v} \quad (4)$$

For the window size as above, we require a 11 bit right-shift for division as in (5).

$$\text{Shift: } \mu_X = \mu_X \gg p * q \quad (5)$$

The computation of pixel mean, is used in the cluster initialization stage, as described next.

3.3 Clustering in Perception Space

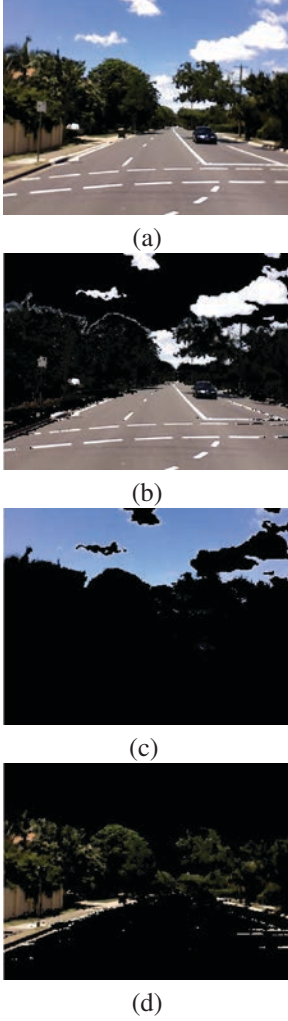


Figure 6. Original image (a). Training clusters (K=3) in (b-d).

The pre-filter stage, allows for cluster initialization using the dominant terrain color. Subsequently, we operate on a perception sub-space, $T \subset X$. Individual pixels, $t(u, v) \in T$ correspond to bit-shifted versions of $x(u, v)$. We define a scalar shift operator b to obtain $t(u, v) \leftarrow x(u, v) \gg b$, which is a bit-shifted version of x . Consequent the mapping on T is defined by $X \xrightarrow{b} T$. This pre-processing before clustering, reduces the search space from 2^{24} for the standard 24 bit representation of color images, to $2^{(24-3b)}$ unique combinations of colors. In our implementation, we set b to 4. This choice led to substantial improvement in computation throughput, while maintaining sufficient resolution in color space, with distinguishing features.

K-means clustering (Duda and Hart[1]) is performed on the m training examples. Specifically, the training data set is $\{t^{(1)}, t^{(2)}, \dots, t^{(m)}\} \in T$.

The cluster initialization is performed for each of the K cluster centroids $\{\mu_1, \mu_2, \dots, \mu_K \in \mathbb{R}^3\}$. μ_1 is set to the mean μ_X , and each of the rest μ_k are randomly initialized.

The cluster assignment step, for the i^{th} training example is indexed by (6).

$$C^{(i)} \leftarrow k \text{ that minimizes } \|t^{(i)} - \mu_k\| \quad (6)$$

Subsequently, the move centroid step is performed using (7).

$$\mu_k \leftarrow \frac{1}{|C_k|} \sum_{i \in C_k} t^{(i)} \quad (7)$$

The cost function which is minimized is given by (8).

$$J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K) = \frac{1}{m} \sum_{i=1}^m \|x^{(i)} - \mu_{c^{(i)}}\| \quad (8)$$

At this stage of the pipeline, we have K -color clusters, which become training models representative of navigable terrain, as shown in figure 6. We use this data, and a past history of the color clusters, to update the terrain model.

3.4 Horizon detection

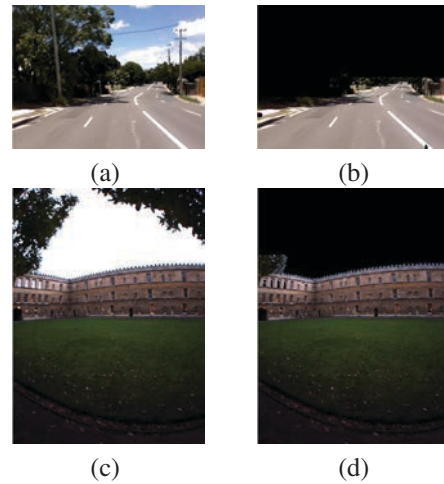


Figure 7. Horizon detection, performed on the original images (a),(c). Filtered results after horizon removal shown in (b),(d).

To update the model, we score individual pixels in the visual field, based on computation of a distance metric from each cluster center. To reduce the number of computations, pixels which comprise of the sky region, are removed using a horizon detection scheme. Horizon detection, is performed at the coarse scale by thresholding pixel intensity values. The variance of the dominant color model, which corresponds to the ROI, is computed using (9).

$$\sigma_X^2 = \sum_{(u,v) \in X} (z_{u,v} - m_X)^2 P(z_{u,v}) \quad (9)$$

We apply a thresholding scheme for pixels falling within two standard deviations of the mean in the active window, denoted by $g_{horizon}$.

$$g_{horizon}(u, v) = \begin{cases} 1, & \text{if } z_{u,v} \in (-2\sigma_{S_X}^2, 2\sigma_{S_X}^2). \\ 0, & \text{otherwise.} \end{cases} \quad (10)$$

The horizon in this case is defined as the row of pixels in $g_{horizon}$ which have the largest occurrence of zeros, or minimal sum. Formally, the horizon H , for an $n \times m$ image is defined as a collection of pixels at row u_h : $\{h(u_h, 1), h(u_h, 2), \dots, h(u_h, m)\} \in H$.

u_h is obtained using (11).

$$u_h \leftarrow u_i \text{ that minimizes } \sum_{u_i, v=1}^m g_{horizon}(u_i, v) \quad (11)$$

This scheme is limited to instances where a clear horizon is defined in color space, it fails when the rover is turning. In such scenarios, u_h gets assigned to null, and the horizon detection stage is skipped in the overall pipeline.

$g_{horizon}$ is updated according to (12).

$$g_{horizon}(u, v) = \begin{cases} 1, & \text{if } u < u_h. \\ 0, & \text{otherwise.} \end{cases} \quad (12)$$

Figure 7 shows images, after the horizon removal has been performed. We now have our training data ready, to update the Mixture of Gaussians.

Individual pixels in the visual field are scored based on the Mahalanobis distance from the learned models. At this stage, we have K color clusters, characterized by their mean μ , covariance Σ , and number of pixels in the cluster, or cluster mass, ω . In addition to the K training models, we have N

learned models, which incorporate the past history of dominant terrain color, with $N > K$.

Initially, all the N learned models are null. Model update occurs, if there is an overlap between the k^{th} training model, and n^{th} learned model, according to (13) (where subscripts T denotes one of the K training models, and L denotes a learned model N). These equations are reproduced from research by [7], with modification to reduce computation cost.

$$(\mu_L - \mu_T)^T (\Sigma_L + \Sigma_T)^{-1} (\mu_L - \mu_T)^T \leq 1 \quad (13)$$

We restrict the covariance matrix to be diagonal, and save computation time associated with full covariance matrix inversion.

When the condition in 13 is satisfied, the learned model is updated. The update equations are as in (14)–(16):

$$\mu_L \leftarrow \frac{\mu_L \omega_L + \mu_T \omega_T}{\omega_L + \omega_T} \quad (14)$$

$$\Sigma_L \leftarrow \frac{\Sigma_L \omega_L + \Sigma_T \omega_T}{\omega_L + \omega_T} \quad (15)$$

$$\omega_L \leftarrow \omega_L + \omega_T \quad (16)$$

3.5 Scoring pixels in perception space

If the current training mode does not match any of the learned models, as per (13), then the model update is performed using (17).

$$\left. \begin{aligned} \mu_L &\leftarrow \mu_T \\ \Sigma_L &\leftarrow \Sigma_T \\ \omega_L &\leftarrow \omega_T \end{aligned} \right\} \text{that minimizes } \omega_n. \quad (17)$$

This summarizes the scoring mechanism and learning scheme, to incorporate a history of terrain color in the perception scheme. Figure 8 shows learning dominant color models, in an indoor environment.

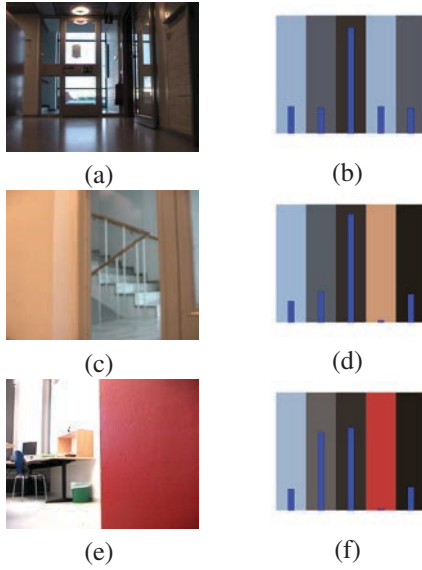


Figure 8. Learning terrain models in the indoor environment. Original images (a),(c),(e), dominant terrain colors overlaid with model mass are shown in (b),(d), and (e). Model update occurs by replacing the cluster with lowest mass, with new training data. Images look best, when viewed in color.

4 Experiments

We validate the proposed scheme on several standard databases, across indoor and outdoor environments.

Figure 9 is representative of typical test scenarios. Additionally, we also compare the proposed approach to labeling training data by [7], using the Microsoft[®] Kinect[™] Sensor, as our range finder. A brief description of the datasets and experimental setup follows. Figure 10 shows our experimental rover platform on which we implemented our vision system.

4.1 Description of Datasets

We chose to validate terrain learning on heterogeneous illumination conditions, varying obstacle field configurations, and diverse environment dynamism. The standard data bases include monocular urban road sequences, from the RAT-SLAM field experiments as described by [4]. The dataset,

comprises of several miles of urban road, and includes moving vehicles and pedestrians during different times of the day.

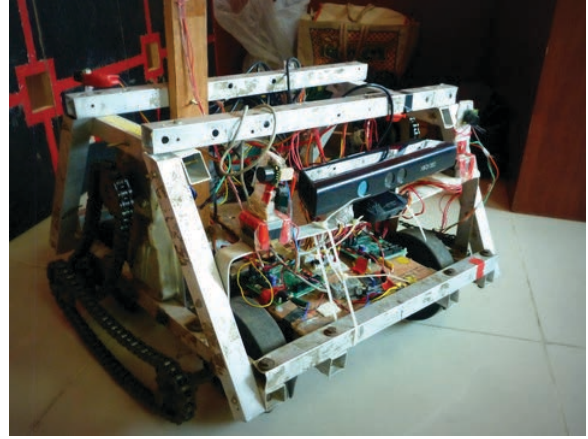


Figure 10. The ‘Freelancer,’ a custom built robotic platform, equipped with a camera and proximity sensors, is used for field testing our perception scheme.

The New College Dataset, is a collection at Oxford, for vision and mobile robot research. The dataset comprises of images captured from an omnidirectional camera, a laser ranger, and stereo imagery, over the college’s grounds and neighboring parks. We validate our algorithm using the vision data, captured by the Ladybug platform. The dataset is described in [17].

For validation in the indoor environment, we run tests on the standard IDOL2(Image Database for rObot Localization) database. A formal description is available by [14]. The dataset incorporates several trials under varying illumination conditions: cloudy, sunny, and at night. Additionally, the data is scattered with instances of moving people.

The above datasets span structured outdoor and indoor environments. Additional experiments, in unstructured environments were also conducted. The dataset, populated at The Cubbon Park, in Bangalore, offered several navigable terrain profiles. We demonstrate the efficacy of the method, on a walkway, in a bamboo forest, and on dirt and grass terrain at the park. We describe the outcome of the experiments, based on absolute pixel misclassification error in the following section.



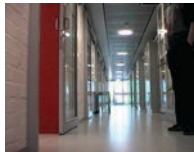


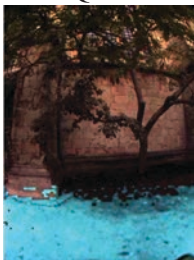

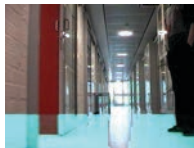












New College Dataset, Oxford	St. Lucia Suburb (RAT-SLAM Dataset)	IDOL2 Dataset, KTH	Cubbon Park Dataset	Urban Road and Apartment Trials
–Outdoor –Structured –Low Dynamism –Cloudy weather	–Outdoor –Structured –High Dynamism –Clear skies	–Indoor –Structured –Low Dynamism –Varying lighting	–Outdoor –Unstructured –Static –Varying lighting	–Outdoor –Structured –Varying dynamism –Varying lighting
 (a) Tree at the Quad	 (b) In urban traffic	 (c) Moving people	 (d) Grassy terrain	 (e) Parking lot
 (f) Free space	 (g) Car as obstacle	 (h) Person avoided	 (i) Free space	 (j) Free space
 (k) At the Quad	 (l) Shadow on road	 (m) Cluttered indoor	 (n) Sunlight on dirt terrain	 (o) Traffic Intersection
 (p) Learns grass terrain	 (q) Shadow is navigable	 (r) Free space	 (s) Learns patchy terrain	 (t) Delineates road, misclassified white cars

Figure 9. The figure shows the spectrum of experiments used to validate the proposed method, on standard datasets. The variations in test conditions include illumination changes, environment dynamism, and structure. The tests were conducted off-line on standard datasets, and on-line for the field and parking lot tests. Images look best, when viewed in color

4.2 Training Stage Validation

In this section we describe comparing our method, with the standard implementation using a laser range finder, by [7]. The pixels, chosen as training samples are delineated navigable, by parsing through range finder data. The Kinect sensor, captures three dimensional scene depth, upto 10 meters from the current position of the sensor. The point cloud, is a collection of samples, $\{d_1, d_2, \dots, d_m\} \in D$, where $d_i \in \mathbb{R}^3$. d_i denotes the Cartesian coordinates of a point in space, with reference to camera coordinates at the origin.

To obtain training samples, the authors [7], model the height difference uncertainties between samples as a temporal Markov chain. The details of the implementation, that uses a laser scanner, and 6 degree of freedom pose estimator ([19]).

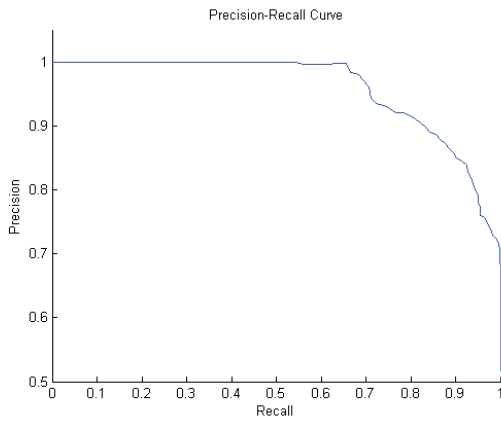


Figure 11. Precision-Recall Curve for the training stage, with true positives $\in X_D \cap X$ and false positive $\in X$.

We simplify the procedure, by thresholding the point cloud, to restrict data to near-range navigable terrain. Formally, we define an threshold d_{th} computed using (18). The choice of the exponential function is motivated from our previous investigation into learning from raw-sensor depth ([16]), where the parameters a and b are learned at system initialization.

$$d_{th} = ae^{bz} \quad (18)$$

Point cloud filtering is performed to obtain pixel locations, $(x, y) \in X_D$, of training samples by subspace decomposition. The decomposition is performed according to 19.

$$X_D = \{(x, y) \mid (x, y, z) \in D \text{ and } z < d_{th}\} \quad (19)$$

Subsequently, we compare pairwise location correspondence using X_D and pre-filtered pixel locations, X in the ROI. Figure 11 shows the precision-recall curve, at d_{th} . Our experimental results suggest that at the worst case, the position difference between pixels in X and X_D is limited to 30.

Table 1. Average computation time at different pipeline stages. The averages, standard deviation, and max time is calculated per frame update. The results are compiled for the Cubbon Park dataset, for a 5 level octave pyramid, bit-shift $b = 3$, $N = 5$, and $k = 3$.

Pipeline Stage	Average time (s)	Standard Deviation (s)	Max Time(s)
Octave Pyramid	0.008133	0.054102	0.151172
Bit-shift level	0.000095	0.000018	0.000119
Pre-Filter Stage	0.008524	0.000502	0.008974
Pixel Clustering (Training Stage)	0.000236	0.001347	0.000265
Pixel Clustering (Model Update)	0.006710	0.036943	0.009510
Horizon Detection	0.000326	0.002104	0.009565

5 Results

Our experimental results are summarized in figure 9. Statistics like worst case pixel misclassification, is found in our previous work ([15]). The change in frame throughput, using the octave pyramid, bit-shifted perception space, and a combination of both is shown in the graphs below.

The performance curves (figures 12, 13 and 14), show increased throughput by forming the octave pyramid. The time for individual operations in the pipeline is included in table 1. The saving in computation time, is less pronounced when using only the bit-shifted perception subspace. A combination of the two schemes, allows a reduction of 0.8 seconds, at $b = 1$.

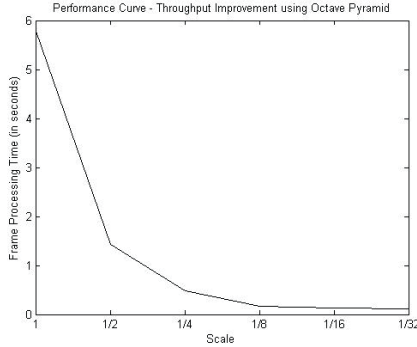


Figure 12. Exponential reduction in processing time, is seen at coarse levels of the octave pyramid.

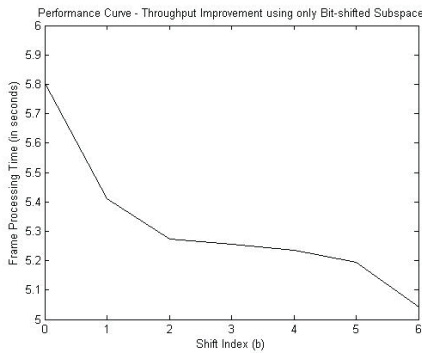


Figure 13. The gain in throughput increases till $b = 3$, and saturates beyond that value. Choosing $b > 5$ produces resultant images of poor visual quality.

6 Conclusion

In this paper, we have described a computationally inexpensive algorithm to identify navigable terrain, using a color based clustering approach. Additionally, our primary sensing modality, is a single consumer grade web-camera. The proposed monocular vision scheme, is shown to be as effective in delineating training samples, as multi-sensor data fusion from a laser ranger. Table 2 shows the mean error rate of the classifier, based on the experiments we conducted.

The scheme is completely implemented using an integer data-path; employs bit-shifts and accumulate operations. Our perception method, has exponentially faster per frame throughput, compared

to the original approach. The computation time is reduced by orders of magnitude, by learning dominant terrain color in bit-shifted perception subspace, at the coarse level of an octave pyramid.

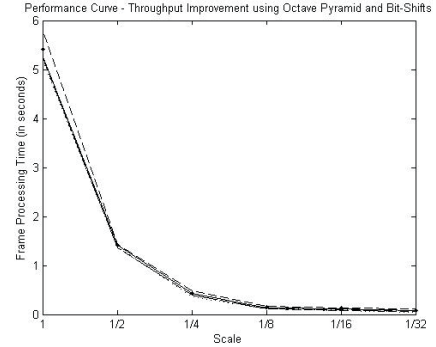


Figure 14. On combining the two schemes, it is evident from the figure, that operating at the coarse level allows for significant reduction in computation overhead. Additionally, a boost (of 0.87 seconds) is obtained by bit-shifting the perception subspace, using $b > 2$. An overall boost in performance of upto 60x is achieved.

Limitations of the monocular vision framework, lies in misclassifying objects which are similar to the dominant terrain color. Such corner cases violate our assumption of heterogeneous free-space and obstacle color. Specific examples are evident in the case of gray cars misclassified as traversable, on the urban road tests. We acknowledge this limitation in our perception scheme, which makes a strong case for multi-sensor data fusion. The fusion module would operate on a higher hierarchical layer, taking into account the terrain learnt using a single camera, and there are gains to be realized here.

We tested monocular vision based terrain learning in diverse terrain types. Validation spanning variations in obstacle field configuration, illumination conditions, and environment structure, shows the feasibility of our approach.

7 Acknowledgements

The authors gratefully acknowledge Professor Koshy George, Director at the PES Centre for Intelligent Systems, for his feedback and suggestions.

Table 2. Mean error rate of the self-supervised learning algorithm on different datasets.

Dataset	(Outdoor/ Indoor)	Description (Structured/ Unstructured)	(Static/ Dynamic)	Unmodified Algorithm	Image Pyramid (scaled 1/32x Original)	Image Pyramid and Bit-shifts (b = 3)
Standard Test Datasets						
New College Dataset (Oxford Mobile Robotics)	Outdoor	Structured	Static	5.88 %	6.87%	7.24%
St. Lucia Loop (Rat-SLAM Dataset)	Outdoor	Structured	Dynamic	12.13 %	12.55%	12.67%
IDOL2 Dataset	Indoor	Structured	Dynamic	5.5 %	7.32%	7.76%
Custom Test Datasets						
Apartment Complex	Outdoor	Structured	Static	11.97%	12.45%	13.66%
Urban Road	Outdoor	Structured	Dynamic	15.34 %	16.12%	16.83%
University Campus	Outdoor	Structured	Dynamic	10.67%	11.31%	12.98%
Cubbon Park Dataset	Outdoor	Unstructured	Dynamic	15.56 %	17.88%	18.45%

References

- [1] R.O. Duda, P.E. Hart, and D.G. Stork. *Pattern Classification*, John Wiley and Sons, 2nd Edition, New York 2001.
- [2] *J. Field Robot.*, 25(9), 2008.
- [3] Max Bajracharya, Andrew Howard, Larry H. Matthies, Benyang Tang, and Michael Turmon. Autonomous off-road navigation with end-to-end learning for the lagr program. *Journal of Field Robotics*, 26(1):3–25, 2009.
- [4] David Ball, Scott Heath, Janet Wiles, Gordon Wyeth, Peter Corke, and Michael Milford. Open-ratslam: an open source brain-based slam system. *Autonomous Robots*, 34(3):149–176, 2013.
- [5] Martin Buehler. Summary of dgc 2005 results: Editorial. *J. Robot. Syst.*, 23(9):659–660, September 2006.
- [6] P.J. Burt and E.H. Adelson. The laplacian pyramid as a compact image code. *Communications, IEEE Transactions on*, 31(4):532–540, 1983.
- [7] Hendrik Dahlkamp, Adrian Kaehler, David Stavens, Sebastian Thrun, and Gary R. Bradski. Self-supervised monocular road detection in desert terrain. In Gaurav S. Sukhatme, Stefan Schaal, Wolfram Burgard, and Dieter Fox, editors, *Robotics: Science and Systems II, August 16-19, 2006. University of Pennsylvania, Philadelphia, Pennsylvania, USA*. The MIT Press, 2006.
- [8] Bob Davies and Rainer Lienhart. Using cart to segment road images. pages 60730U–60730U–12, 2006.
- [9] G.N. DeSouza and A.C. Kak. Vision for mobile robot navigation: a survey. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(2):237–267, 2002.
- [10] John A. Grant, Matthew P. Golombek, John P. Grotzinger, Sharon A. Wilson, Michael M. Watkins, Ashwin R. Vasavada, Jennifer L. Griffes, and Timothy J. Parker. The science process for selecting the landing site for the 2011 mars science laboratory. *Planetary and Space Science*, 59(1112):1114 – 1127, 2011. Geological Mapping of Mars.
- [11] Raia Hadsell, Pierre Sermanet, Jan Ben Ayse Erkan, and Marco Scoffier. Learning long-range vision for autonomous off-road driving. *Journal of Field Robotics*, pages 120–144, 2009.
- [12] Wes Huang, Greg Grudic, and Larry Matthies. Editorial. *Journal of Field Robotics*, 26(1):1–2, 2009.

- [13] L.D. Jackel, Douglas Hackett, Eric Krotkov, Michael Perschbacher, James Pippine, and Charles Sullivan. How darpa structures its robotics programs to improve locomotion and navigation. *Commun. ACM*, 50(11):55–59, November 2007.
- [14] J. Luo, A. Pronobis, B. Caputo, and P. Jensfelt. Incremental learning for place recognition in dynamic environments. In *Proc. IROS07*, 2007.
- [15] P. Mishra and A. Viswanathan. Computationally inexpensive labeling of appearance based navigable terrain for autonomous rovers. In *Intelligence in Vehicles and Transportation Systems (CIVTS), 2013, IEEE Symposium on*, pages 88–92, April 2013.
- [16] P. Mishra, A. Viswanathan, and A. Srinivasan. A supervised learning approach to far range depth estimation using a consumer-grade rgb-d camera. In *Electronics, Computing and Communication Technologies (CONECCT), 2013 IEEE International Conference on*, pages 1–6, 2013.
- [17] M. Smith, I. Baldwin, W. Churchill, R. Paul, and P. Newman. The new college vision and laser data set. *The International Journal of Robotics Research*, 28(5):595–599, May 2009.
- [18] Richard Szeliski. *Computer vision: algorithms and applications*. Springer, 2010.
- [19] S. Thrun, M. Montemerlo, and A. Aron. Probabilistic terrain analysis for high-speed desert driving. In *Proceedings of Robotics: Science and Systems*, Philadelphia, USA, August 2006.
- [20] Sebastian Thrun, Mike Montemerlo, Hendrik Dohlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, Kenny Lau, Celia Oakley, Mark Palatucci, Vaughan Pratt, Pascal Stang, Sven Strohband, Cedric Dupont, Lars-Erik Jendrossek, Christian Koelen, Charles Markey, Carlo Rummel, Joe van Niekerk, Eric Jensen, Philippe Alessandrini, Gary Bradski, Bob Davies, Scott Ettinger, Adrian Kaehler, Ara Nefian, and Pamela Mahoney. Stanley: The robot that won the darpa grand challenge: Research articles. *J. Robot. Syst.*, 23(9):661–692, September 2006.
- [21] Edward Tunstel and Ayanna Howard. Sensing and perception challenges of planetary surface robotics, 2002.
- [22] Christopher Urmson, Joshua Anhalt, Hong Bae, J. Andrew (Drew) Bagnell, Christopher R. Baker, Robert E Bittner, Thomas Brown, M. N. Clark, Michael Darms, Daniel Demitrish, John M Dolan, David Duggins, David Ferguson, Tugrul Galatali, Christopher M Geyer, Michele Gittleman, Sam Harbaugh, Martial Hebert, Thomas Howard, Sascha Kolski, Maxim Likhachev, Bakhtiar Litkouhi, Alonzo Kelly, Matthew McNaughton, Nick Miller, Jim Nickolaou, Kevin Peterson, Brian Pinnick, Ragunathan Rajkumar, Paul Rybski, Varsha Sadekar, Bryan Salesky, Young-Woo Seo, Sanjiv Singh, Jarrod M Snider, Joshua C Struble, Anthony (Tony) Stentz, Michael Taylor, William (Red) L. Whittaker, Ziv Wolkowicki, Wende Zhang, and Jason Ziglar. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics Special Issue on the 2007 DARPA Urban Challenge, Part I*, 25(8):425–466, June 2008.
- [23] Shengyan Zhou, Junqiang Xi, Matthew W. McDaniel, Takayuki Nishihata, Phil Salesses, and Karl Iagnemma. Self-supervised learning to visually detect terrain surfaces for autonomous robots operating in forested terrain. *J. Field Robot.*, 29(2):277–297, March 2012.