Online Learning of Multiple Perceptual Models for Navigation in Unknown Terrain

Greg Grudic, Jane Mulligan, Michael Otte, and Adam Bates

University of Colorado at Boulder, Boulder CO, USA
`Jane.Mulligan@Colorado.edu`

Summary. To navigate efficiently in new terrain an autonomous vehicle should be able to observe and learn perceptual models for identifying traversable surfaces and obstacles, to allow steering and planning in the near and far field. As the robot passes through the environment however, the appearance of ground plane and obstacles may vary, for example in open fields versus tree cover or paved versus gravel or dirt tracks. In this paper we describe a working robot navigation system based primarily on colour imaging, which learns sets of models online as it moves through the environment choosing whether to apply current models, discard inappropriate models or acquire new ones. These models operate on complex natural images and because they are acquired and used as the robot navigates, learning and evaluation must be possible in real time.

1 Introduction

The system described in this paper operates on an autonomous wheeled vehicle designed for the DARPA Learning Applied to Ground Robots (LAGR) program (Fig. 1). The goal of the program is to apply Machine Learning techniques to autonomous navigation in unknown, unstructured terrains [4].

Fig. 1. DARPA LAGR platform.

The LAGR platform is equipped with Stereo camera systems, but camera resolutions and geometry constrain the resolution and discrimination of stereo depth data making it useful to about 10m. As a result considerable effort by program participants has been devoted to using colour and texture for classification of "traversable terrain" and obstacles in the far field. Frequently a standard Learning approach such as Neural Nets, SVM or Decision Trees [2, 3, 1] is applied to build a single model for the terrain at a particular test site. This approach has several drawbacks. First the models are large and difficult to adapt online when terrain appearance changes: a single model cannot capture the diversity of terrain necessary for far field navigation. The models require examples of traversable and non-traversable terrain, where often one may only be able to confidently identify traversable regions. Finally, models are blindly applied across the image, even in regions sufficiently dissimilar to the training set to make their results meaningless. The models offer no means to identify where their classifications can be applied with high confidence. In contrast we construct separate density-based models for traversable and non-traversable terrain. As a result of their density structure they respond only to terrain that is similar to the training data from which they were constructed.

Our approach builds a set of models as it traverses a new environment. At each frame the current set of models is applied to the image, and if none explains the data a new model is constructed and added to the set. This allows the robot to pass across varied surfaces avoiding obstacles which differ from one area to the next, as when moving from a rocky open field to a path with tree cover.

The idea of using multiple models to improve adaptive controllers has been studied by Narendra et al. [5], who propose control strategies for dynamic environments exploiting multiple models, switching, and tuning. The main questions when maintaining a set of models for a changing environment are: how to determine when the current models are no longer effective and thus when to construct new models and for large numbers of models choosing which to evaluate at a particular instant.

2 System Overview

The LAGR platform has two "eye" computers devoted to Stereo and image processing. A third computer handles planning and control functions. The main issues for our work are how to construct the image costmaps which classify near and far field regions as traversable and non-traversable, and how these are used to plan appropriate behaviour.

2.1 Combining Cost Maps

The current color vision system utilizes two cost masks, obstacle and ground plane, to create a cost update for the planner. A mask is the same size as the images produced by the robot's cameras and represents the probability of that pixel depicting an obstacle or the ground plane. The reason for using two cost masks is centered on the meaning of non-classified regions in an image. When classifying regions of ground plane using a single class classifier we can only be sure that areas that are classified as in the class are in the ground plane. This doesn't mean that the non classified regions are not in the ground plane. A second mask that classifies obstacle regions is needed. This creates a check for the non-classified regions in each mask. If a point in the ground plane mask has a low probability of being in the ground plane then the decision as to whether that pixel is an obstacle is left to the obstacle classifier. This is also true for the obstacle mask. Therefore the assumption that non-classified pixels are obstacles is only made when the ground plane classification is low and the obstacle classification is high.

The two cost masks must be combined in a meaningful way. We take the difference of the obstacle mask O and the groundplane mask G, of the two masks are taken a new mask $O - G$ is created and scaled from -1 to 1. When the two cost masks disagree, i.e. ground plane mask equals 1 and obstacle mask equals 1 then a 0 is returned. These values are then left to stereo vision to determine. When the cost masks agree, i.e. ground plane mask equals 1 and the obstacle mask equals 0 then a -1 is return. This means that when sent to the planner the cost in that location will be reduced. When the planner is sent a positive value the cost in that location is increased.

Once the two cost masks are obtained and their difference is found a transformation from image coordinates to robot coordinates must be made. The transformed difference will be sent to the planner. Since the camera properties are known, i.e. focal length and center, and an estimate of the ground plane has been calculated by the stereo vision system, a simple coordinate transform computes pixels XY coordinates in the robot frame of reference. Once each pixel's transformed coordinates are found they are binned into 20×20 cm bins. The average of each bin is found and normalized by the total number of pixels transformed, thus bins close to the robot are weighted higher than bins further from the robot. This new cost map update is then sent to the planner.

2.2 The Planner

The planner responsible for accumulating data provided by the cameras organizing it, and then using the data to compute a feasible route to a predetermined goal. Usually, the goal is a GPS coordinate, and the route is chosen to minimize some cost function, such as distance, risk, etc. The implementation of the CU robot planner is built on work by Daniel Lee at the University of Pennsylvania. This particular planner is essentially a state machine, where the behavior of the robot in each state S_t is dictated by the events that occurred in the previous state S_{t-1}, as well as a global state G. Information from G is usually related to outside sources (bumper hits, human input, messages from the eyes, etc.), while information from M pertains more to tasks that the planner has scheduled for itself (calculation of a new path, movement of a certain distance, etc.). G also assumes responsibility for placing cost information from the eyes into a perpetual global cost map.

The cost map is a 2D birds eye view of the area that the robot has previously sensed and/or can currently perceive. In practice, the map is stored in a 2-dimensional array that maintains the relative spatial locations of cost values, with each piece of the map being l long on each side. The information that is placed onto the map arrives in two forms: stereo depth data, and data obtained by running image attributes through machine learning models. Depending on the navigation task defined at runtime (attempting to avoid water or go through tall grass), this information can be weighed differently. It can also be combined with the information already in the global map in a variety of ways, including: replacement, incrementation, weighted averaging, or some hybrid. Regardless of source or combinational scheme, the costs in the cost map always fall on a spectrum of belief about the navigability of their associated regions in the map. High cost means that an area is less navigable than low cost.

Most of the planner's time is spent in StatePlan, the state responsible for finding a path from the current location to the goal. StatePlan searches for a path using $A*$ search with a cost function defined at runtime. For the purposes of $A*$, a path is composed of a set of adjacent pieces of map. For most robot navigation problems it is desirable that the cost function reflect some combination of navigability and distance. A principled way to optimize over these two variables is to represent low navigability as a force that the robot must overcome; and then minimize the amount of work that the robot must do in order to reach the goal. The function to minimize is given by:

$$\textit{Navigable Work} = C_1 D_1 + C_2 D_2 + \ldots + C_{\text{goal}} D_{\text{goal}}$$

Where C_i and D_i represent the cost and length of path segment i, respectively.

Once a path to the goal has been found, the planner interacts with the low level motors and servos to navigate the robot along the path until there is a change in state, or a predetermined amount of time has passed. In the latter case, the planner reenters StatePlane, and the process repeats itself.

3 The Learning Framework

3.1 Fast Density Models

The approach taken is to build two types of density models - one for identifying traversable terrain and the second for non-traversable terrain. The traversable terrain density models are constructed by sampling image regions that are associated with traversable terrain, and similarly for non-traversable terrain density models.

A number of powerful techniques have been proposed in the Machine Learning and Statistics communities for density estimation [6]. However, although these techniques can be effective in high dimensional spaces (such as those addressed in this paper), invariably these techniques are too computationally intensive to be applied to real-time robot navigation applications. In this paper, we present an efficient framework for density estimation that is suitable for real-time terrain classification. Our framework builds many small, fast density models, which are combined to produce a final density model for the entire image[1].

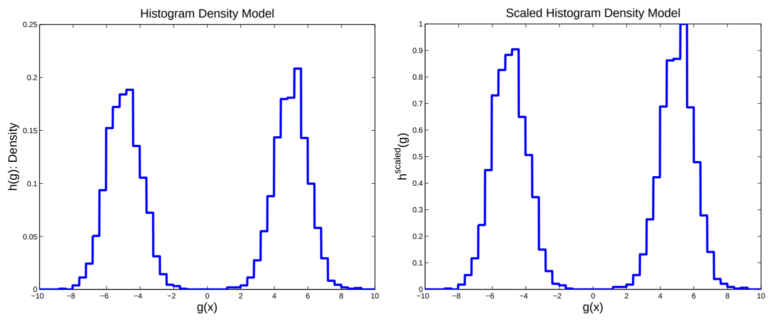
Assume a set of N examples, each of dimension d, extracted from an image region that is known to be traversable. We symbolize these as $\{\mathbf{x}_1, ..., \mathbf{x}_N\}$, where each $\mathbf{x}_i \in \Re^d$ for all $i \in \{1, ..., N\}$ is a column vector representing a set of features extracted from an image. Next, the NULL space and the Basis space (i.e. the Principle Components ordered according to relevance) of this data is computed using Singular Value Decomposition. We symbolize the Null space and Basis space matrices as N_{tr} and B_{tr} respectively (the underscore tr here indicates that a traversable model is being constructed). Note that N_{tr} has d rows and m columns, B_{tr} has d rows and k columns, where $k + m = d$ (not that k and d are determined in a standard way based on the floating point precision of the CPU). The NULL space is used to identify when a new sensor readings do NOT fall within the scope of the model (i.e. if any new sensor reading falls in the null space of the model, the model outputs 0). Therefore, given some new feature set $\mathbf{x}$, the model outputs 0 if $\|N_{tr}^{'}\mathbf{x}\| < \varepsilon$, where ε is determined by the floating point of the CPU.

Next the remaining data is projected in the Basis space B_{tr}, giving by $\mathbf{b}_i = B_{tr}^{'}\mathbf{x}_i$, for all $i = 1, ..., N$, where each $\mathbf{b}_i \in \Re^k$. This data is and divided into two approximately equal halves - one set is the training set $\{\mathbf{b}_1^t, ..., \mathbf{b}_{N_t}^t\}$, and the other is the validation set $\{\mathbf{b}_1^v, ..., \mathbf{b}_{N_v}^v\}$, where t refers to training data, v refers to the validation data, N_t refers to the number in the training set, and N_v refers to the number in the validation set (note that $N_v + N_t = N$). A linear function is then defined $g_{\text{tr}}(\mathbf{x}) = \mathbf{p}^{'}\mathbf{x}$, where $\mathbf{p}$ is the first column of the Basis space B_{tr}, corresponding to the largest eigenvalue of the data PCA space, and thus the coordinate which contains the greatest data range. Next $g_{\text{tr}}(\mathbf{x})$ is converted to a density model using a one dimensional histogram model called $h_{\text{tr}}(g_{\text{tr}}(\mathbf{x}))$. This histogram density model is constructed by choosing bin size that maximize the negative log likelihood of the validation set $\{\mathbf{b}_1^v, ..., \mathbf{b}_{N_v}^v\}$ (in the first dimension). By maximizing the negative log likelihood on the validation data, we ensure that the resulting density model is most representative of the training data. A typical histogram density model is depicted in Figure 2(a). The remaining $k - 1$ dimensions are used to find the minimum and maximum values that $\{\mathbf{b}_1^t, ..., \mathbf{b}_{N_t}^t\}$, and $\{\mathbf{b}_1^v, ..., \mathbf{b}_{N_v}^v\}$ have along these coordinates - any new features falling outside these ranges result

[1] Two such final models are produced, one for traversable terrain and the other for non-traversable.

in the model outputting 0. Finally, if training data is available from the non-traversable class, it is used to adjust the density model $h_{\mathrm{tr}}(g_{\mathrm{tr}}(\mathbf{x}))$ such that the traversable training data is always more probable than the non-traversable data in each bin of the histogram. This ensures that the models are as selective as possible in predicting traversable regions. This algorithm constitutes an efficient and effective framework for bounded density estimation for robotics applications.

The exact same procedure is used to construct density models for non-traversable terrain, giving $g_{\mathrm{ntr}}(\mathbf{x})$ and $h_{\mathrm{ntr}}(g_{\mathrm{ntr}})$.

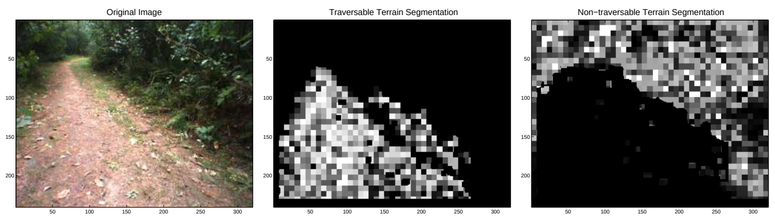


(a) Histogram Density Model (b) Scaled Histogram Density Model for Image Segmentation

Fig. 2. Producing Density Models For Image Classification.

3.2 Converting a Density Model to a Segmented Image

Every model $h_{\mathrm{ntr}}(g_{\mathrm{ntr}})$ is used to segment out images of into traversable terrain, and, similarly, every model $h_{\mathrm{tr}}(g_{\mathrm{tr}})$ is use segment out images into traversable terrain. This is done by scaling the density models to range between 0 and 1 as in Figure 2(b), which is obtained by taking the density model in Figure 2(a) and dividing by its maximum value. Once this is done, every segment of the image can be classified as shown in Figure 3. Figure 3(a) shows the original image, Figure 3(b) shows the resulting segmentation after passing the image through the traversable terrain scaled histogram model $h_{\mathrm{tr}}(g_{\mathrm{tr}})$, and Figure 3(c) shows the resulting segmentation after passing the image through the non-traversable terrain scaled histogram model $h_{\mathrm{ntr}}(g_{\mathrm{ntr}})$. A key property of these scaled density models is that, in every region where the image is dark, the density models have no opinion on whether the region is traversable or non-traversable. This implies that the model is able to predict when image regions or too dissimilar from the data used to construct it, and therefor no predictions can be made using it. As discussed below, this property gives a formal framework for deciding when new models must be added, which models are most appropriate for a given image, and for combining multiple models.


Original Image
Traversable Terrain Segmentation
Non−traversable Terrain Segmentation
50
100
150
200
250
300

(a) Original Image (b) Traversable Segmentation (c) Non-traversable Segmentation

Fig. 3. Using the Scaled Density Models to segment traversable and non-traversable image regions. Image brightness is proportional to confidence in segmentation. The data used to construct the segmentation models was obtained using stereo in the near range.

3.3 Learning Multiple Models

In the proposed framework, we construct a set of traversable terrain models $\{h^1_{\text{tr}}(g^1_{\text{tr}}), ..., h^{N_{tr}}_{\text{tr}})(g^{N_{tr}}_{\text{tr}})\}$, as well as a set of non-traversable models $\{h^1_{\text{ntr}}(g^1_{\text{ntr}}), ..., h^{N_{ntr}}_{\text{ntr}}(g^{N_{ntr}}_{\text{ntr}})\}$. Each model represents a different type of terrain that the robot encounters as it navigates, and all are constructed in real time. The decision on how many models are need is made automatically- whenever the current set of models do not explain the observations made by stereo, new models are added. Thus, if the current set of traversable models do not label a traversable portion of the image as being traversable, a new traversable model is constructed. Similarly, if a region in the image is identified as non-traversable, and the current non-traversable do no label this region as such, then a new non-traversable model is constructed to account for this new terrain type. Thus, as the robot traverses over a variety of different terrains, more models are needed, resulting in potentially thousands of models.

3.4 Choosing Model Subsets

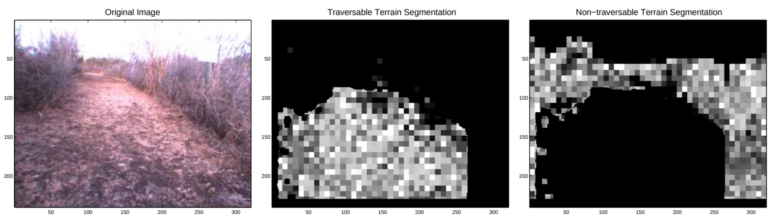
Each density model is representative of a particular terrain type, and as the type of terrain changes over time, every model may not be appropriate for every image. In particular, if a model outputs zero everywhere in an image, then it was constructed from a terrain type that does not exist in the image, and therefore should not be used. Similarly if a traversable model has a non-zero response in an image region that is in fact NOT traversable, then it should not be used on the current image. An identical argument applies to non-traversable models. Thus the models that are applied to any image are sampled from models that are consistent with near field observation in the image. Even after this selection process, there still may be many hundreds of models that are consistent with an image. In order to ensure real-time operation, not all consistent models can be applied to every image. As a result, the consistent models are ranked according to the magnitude of average response over a small random regions of the image. The models with the highest average response are most consistent with the image, while the models with the lowest are least consistent. Thus the first N highest ranked models, where N is dictated by real time considerations, are applied to the image.

3.5 Traversable and Non-Traversable Terrain Images From Density Models

Once the subset of traversable models is chosen for application to the image, then creating the final traversable image segmentation is done by taking a maximum over all models, over every region of the image. A similar operation is done for generating the non-traversable image segmentation using the non-traversable model. The justification for this max operation steams from the fact that the models that are applied to an image are all consistent with its near field observations. When there is disagreement between the traversable and non-traversable segmented images, then the robot errs on the side of exploration and labels the inconsistent parts as traversable, leaving stereo to correct this possible miss labeling when the robot gets close enough to the region in question. This type of inconsistency is rare because of the whey the models are chosen, but can none the less occur.

4 Experimental Results

The current set of density models are constructed using 7 by 7 windows in a 320 by 240 RGB normalized image-future models will use the more powerful feature described previously in the paper. Thus, the dimension of each density model constructed is $7 \times 7 \times 3 = 147$. This system has been applied to a wide variety of real tests under the (DARPA LAGR program) in unstructured outdoor environments. The density models where able to segment difficult terrain in the far field (well beyond stereo range) as demonstrated in Figure 4, where the color difference between the non traversable vegetation and the ground is difficult to distinguish. In all the environments tested to date, never more that 47 traversable terrain models, and 43 non-traversable terrain models were required to model the terrain. This demonstrates the flexibility of the individual models. In addition, at each frame up to 20 models where applied (chosen according to the procedure defined Section 3.4) at a frame rate of about 5 frames a second. The current implementation of this system is written in MATLAB, and we anticipate significant speedups in frame rates when these models are implemented in C. It is also worthy to note that such systems that seek to classy terrain beyond stereo range have consistently outperformed robots that only use stereo. The system described here is no exception to this, and, as described in the introduction, has the benefit of fast online learning and quick adaptation to new environments.


Original Image
Traversable Terrain Segmentation
Non−traversable Terrain Segmentation
50
100
150
200
50
100
150
200
250
300

(a) Original Image (b) Traversable Segmentation (c) Non-traversable Segmentation

Fig. 4. Using the Scaled Density Models to segment traversable and non-traversable image regions. Image brightness is proportional to confidence in segmentation. The data used to construct the segmentation models was obtained using stereo in the near range.

5 Conclusion

For autonomous robots traversing unknown terrain, Stereo depth data alone is too "short sighted". Considerable effort has been devoted to learning models of the appearance of traversable terrain to extend near field Stereo observations of ground plane and obstacles to the far field.

In this paper we describe a successful robot system which learns multiple models of obstacle and ground plane online as the terrain varies over the robot's trajectory. We describe an efficient way of choosing the best subset of models for each image, out of a set of models, possibly learned over the robot's lifetime.

The models constructed are density-based and thus respond only to inputs which are sufficiently similar to the data from which they were built. This allows the system to identify novel terrain when its models stop responding, thus triggering the construction of new models. Both model construction, and selection and evaluation are real-time online operations.

Future efforts include finding optimal image features for terrain classification and allowing each model to identify a preferred feature set for the terrain it encodes.

References

1. Jim Albus, Roger Bostelman, Tommy Chang, Tsai Hong, Will Shackleford, and Michael Shneier. Learning in a hierarchical control system: 4d/rcs in the darpa lagr program. *Journal of Field Robotics*, 23(11-12), 2006.
2. M. Happold, M. Ollis, and N. Johnson. Enhancing supervised terrain classification with predictive unsupervised learning. In *Proceedings of Robotics: Science and Systems*, Cambridge, USA, June 2006.
3. Andrew Howard, Michael Turmon, Larry Matthies, Benyang Tang, Anelia Angelova, and Eric Mjolsness. Towards learned traversability for robot navigation: From underfoot to the far field. *Journal of Field Robotics*, 23(11-12), 2006.
4. L. D. Jackel, Eric Krotkov, Michael Perschbacher, Jim Pippine, and Chad Sullivan. The darpa lagr program: Goals, challenges, methodologies, and initial results. *Journal of Field Robotics*, 23(11-12), 2006.
5. Kumpati S. Narendra, Jeyendran Balakrishnan, and M. Kemal Ciliz. Adaptation and learning using multiple models, switching, and tuning. *IEEE Control Systems Magazine*, pages 37–51, June 1995.
6. Pascal Vincent and Yoshua Bengio. Manifold parzen windows. In S. Thrun S. Becker and K. Obermayer, editors, *Advances in Neural Information Processing Systems 15*, pages 825–832. MIT Press, Cambridge, MA, 2003.