2008 IEEE International Conference on
Robotics and Automation
Pasadena, CA, USA, May 19-23, 2008

Online, self-supervised terrain classification via discriminatively trained submodular Markov random fields

Paul Vernaza, Ben Taskar, Daniel D. Lee
GRASP Lab
University of Pennsylvania, Philadelphia, PA 19104
vernaza@seas.upenn.edu, taskar@cis.upenn.edu, ddlee@seas.upenn.edu

Abstract— **The authors present a novel approach to the task of autonomous terrain classification based on structured prediction. We consider the problem of learning a classifier that will accurately segment an image into "obstacle" and "ground" patches based on supervised input. Previous approaches to this problem have focused mostly on local appearance; typically, a classifier is trained and evaluated on a pixel-by-pixel basis, making an implicit assumption of independence in local pixel neighborhoods. We relax this assumption by modeling correlations between pixels in the submodular MRF framework. We show how both the learning and inference tasks can be simply and efficiently implemented—exact inference via an efficient max flow computation; and learning, via an averaged-subgradient method. Unlike most comparable MRF-based approaches, our method is suitable for implementation on a robot in real-time. Experimental results are shown that demonstrate a marked increase in classification accuracy over standard methods in addition to real-time performance.**

I. Introduction

We consider the problem of terrain classification in the setting of mobile autonomous robots. A typical task in this setting consists of having the robot navigate from one point to another without a prior map of the environment. It must typically navigate and avoid obstacles using a sensor that can detect obstacles at close range, such as a stereo camera. Stereo can be employed to classify near terrain as "obstacle" or "ground", and the robot uses that information to navigate safely to its destination.

Recent investigations into this problem have focused on the issue of decreasing the "short-sightedness" of such approaches. It is often the case that the ability of the robot to navigate safely and efficiently is limited by the range of sensors such as stereo, which provide little to no information at long range. This motivates the use of monocular cameras to perform terrain classification. Though they provide no depth information, monocular cameras do provide a rich set of visual features, even at long range.

Since a human can easily perform this type of classification based only on monocular images, it is natural to suppose that a learning approach might be well-suited to the task of classifying terrain based only on monocular features. This is the basic scenario we study in this work.

It is useful to first consider some basic desiderata that have motivated our work. First, it is often impractical for a human to directly provide supervised input for a task such as image classification for a mobile robot. This motivates the use of self-supervision, where one type of sensor is used to train another. In our case, data from a stereo camera is used to train a monocular image classifier. This method results in copious amounts of training data, as stereo provides very dense geometric information at close range. For this reason, a learning method appropriate for the task should be able to handle massive amounts of training data in a scalable, memory-efficient way.

The other major necessity for a learning algorithm in this context is the ability to train and evaluate the classifier online and in real-time, since we will be running it on an autonomous, mobile robot that must recognize obstacles and learn the appearance of new obstacles in real-time.

We will show how recent developments in machine learning and computer vision have inspired our approach to this problem, which addresses all of the desiderata above in addition to admitting a fairly simple implementation. In section II, we summarize previous approaches to the problem, mostly based on independent classification. In section III, we give a brief overview of submodular Markov random fields (MRFs), which allow us to introduce correlations in the classifier. We explain our adaptation of submodular MRFs to the terrain classification domain in section IV. Results that show significant gains over previous approaches are presented in section VI.

II. Previous work

Other solutions to the problem considered here have arisen recently in conjunction in various sorts of autonomous navigation settings. A type of monocular terrain classification was recently used to great effect by the Stanford entry in the recent DARPA Grand Challenge [1]. Thrun et al. describe a self-supervised method for identifying drivable areas in an image based on monocular data. Their approach is based on training a Gaussian mixture model online and using it to classify pixels in an independent way.

Other similar approaches tackle the problem of parameter estimation and sensor fusion via learning in graphical models. One such approach, presented by Sofman et al. [2], models dependencies between features, parameters, and terrain traversability estimates in a Bayes network. Parameters in a log-linear model are estimated via Bayesian linear regression. Our method also performs learning in a graphical model parametrized by log-linear models. However, the critical aspect of our model that differentiates it from this and other similar models is that we do not assume that the estimated

978-1-4244-1647-9/08/$25.00 ©2008 IEEE.

quantities (pixel labels) are independent given the parameters and features.

Although other approaches (especially those based on purely discriminative methods) do not make this assumption as clear, this assumption is usually implied. This the case, for example, for many common approaches based on histograms ([3],[4]) and mixture models ([1],[5]).

One method that does relax this strong independence assumption is described by Wellington et al. in [6]. Their system also performs terrain classification as well as some other estimation tasks about the terrain. They use a similar MRF as part a greater structured graphical model in order to enforce relative smoothness of labelings. However, the clique potentials used in this method are very simple, with a single hand-tuned parameter per class to control how smooth the class is expected to be. Our method uses much a more expressive model that allows (binary) clique potentials that are log-linear in the features, and these log-linear models are learned discriminatively instead of hand-tuned.

Other interesting related work can be found in the computer vision community, where there is a large and growing literature regarding contextual image classification using MRFs and conditional random fields (CRFs). Major differences between that literature and this work lie in the structures of the graphs assumed in addition to the methods used for inference and learning. A common feature of many of these methods is a reliance on approximate inference procedures such as loopy belief propagation [7] or sampling-based methods [8], whereas our model allows for very efficient exact inference. Tree-structured models are also commonly used to make the inference more tractable [9], while this is unnecessary in our model. Finally, learning in such models often optimizes a maximum likelihood criterion, which often also requires some sort of approximate inference [8]. We optimize a discriminative max-margin criterion in this work, for which we only need a MAP estimate, which is easily obtained under the assumptions of our model.

III. Submodular MRFs

We begin with a brief review of submodular MRFs. An MRF is an undirected graphical model designed to compactly encode local correlation structure among a set of random variables. This is accomplished by associating the set of random variables with a graph where we take each random variable to be a node, and edges in the graph represent correlations between random variables. In our case, nodes correspond to pixels, where a binary random variable models the pixel's label. Edges locally join pixels whose labels might be correlated.

We then associate each clique (or image neighborhood) in the graph with a clique potential $\phi_c(\cdot)$. Let us assume we are dealing with discrete random variables. Each clique potential is a function that takes as an argument a vector of possible assignments to the variables in its scope. Like a probability measure, the clique potential assigns high values to probable assignments and low values to improbable assignments. Unlike a probability measure, it usually does not sum to one over the set of possible assignments.

The probability of a complete assignment to all the labels is simply the product of all the clique potentials for that labeling, multiplied by a normalizing constant (also called the partition function). Let y be such an assignment vector. The probability of the assignment is then given as

$$P(y) = \frac{1}{Z}\Pi_{c\in\mathcal{C}}\phi_c(y_c) \tag{1}$$

where $\mathcal{C}$ is the set of cliques, y_c is the portion of the assignment y corresponding to the clique c, and Z is the partition function. Note that general inference in this problem is intractable due to the difficulty in computing Z; however, note that the maximum a posteriori (MAP) estimate of the MRF does not depend on Z, which raises the question of whether finding the MAP estimate is tractable or not.

In this work we consider a specialization of the MRF; namely, MRFs with submodular potentials. For simplicity, we will think of a submodular MRF as restricting the clique potentials by only allowing potentials to be specified for assignments where all the variables in the clique have the same assignment. Furthermore, all such potentials must be greater than or equal to one, and all other potentials are set to one. This model essentially rewards conformity and encourages spatial smoothness if cliques correspond to image neighbors.

The value in this restriction is chiefly that the MAP inference problem for such models is nicely approximated by a simple linear program. If we further restrict the model to the case of unary and binary potentials with binary values as possible assignments, we get the surprising result that the MAP problem can be solved exactly in polynomial time by finding the minimum cut of a certain graph [10]. A minimum cut is a partition of the nodes of a graph into two parts such that the sum of weights of arcs separating them is minimized (see [11] for more information regarding minimum cuts).

Related methods have employed min cuts with great success in areas such as image restoration, image segmentation, and stereo vision. Kolmogorov and Zabih [10] describe an entire class of energy-minimization problems that can be solved via min cuts. As such, extremely efficient algorithms have been developed to solve these problems. As a problem that falls within this class, the binary-label, binary-potential, submodular MRF is thus a very attractive model from a practical point of view.

Although the inference task is therefore efficiently solved, an important question is then how to use submodular MRFs in a supervised learning context; this is the subject studied by Taskar et al. in [12]. Their approach is briefly summarized here. The clique potentials are parametrized by simple log-linear functions of feature vectors; i.e., $\log\phi_c(y_c) = \langle w_{y_c}, x_c\rangle$, where x_c is a feature vector associated with the clique c, and w_{y_c} is a weight vector for the "conformity" assignment y_c. (Remember that assignments containing disagreements are assumed to have potential one; thus the weight vector applies only to "conformity" assignments where all variables in the clique have the same assignment.)

Returning to our specific scenario, node features might consist of per-pixel image features such as color or textures computed via convolution. Arc features would try to capture some factors that would induce a correlation in neighboring pixels, such as the presence (or absence) of a particular color combination or edge in the image.

Learning then consists of finding a weight vector that will force the MAP estimate of the MRF to be equal (or close) to a desired assignment given a set of feature vectors associated with the cliques in the MRF. To make this a well-posed problem, Taskar introduces the concept of a margin that is defined in a way similar to the margin in support vector machines. In this case, the margin is defined such that the log-probability of the desired assignment is greater than the log-probability of some other assignment by an amount that scales bilinearly in the margin and the number of disagreements between the desired assignment and the other assignments. This inequality must hold for all assignments. The problem of finding the weight vector is thus changed to the problem of maximizing the margin subject to all of these inequality constraints.

It is ultimately shown that the inference problem can be approximated by an efficient linear program, and the learning problem can be formulated as an efficient quadratic program. This concept is the starting point for this work.

IV. Efficient inference and learning

We now describe our adaptation of the submodular MRF framework to the problem of binary image classification. We consider only submodular MRFs with binary potentials and binary labels, in which the inference problem is known to be tractable, as described before. We begin by developing some notation for the MAP inference problem in a way that parallels [12], though simplified for the case of binary potentials and labels.

A. MAP inference

In the MAP inference problem, we wish to maximize the log-probability of the MRF over all joint labelings y in the set of valid labelings, $\mathcal{Y}$. We express this in the following way, where $L(y) = \arg\max_y \log P(y)$:

$$\begin{aligned}
L(y) &= \arg\max_{y\in\mathcal{Y}} \log \Pi_{c\in\mathcal{C}} \phi_c(y_c) - \log Z \\
&= \arg\max_{y\in\mathcal{Y}} \sum_{i\in\mathcal{N}} \log \phi_i(y_i) + \sum_{ij\in\mathcal{E}} \log \phi_{ij}(y_{ij}) \\
&= \arg\min_{y\in\mathcal{Y}} -(\sum_{i\in\mathcal{N}} \langle w_0, x_i\rangle(1-y_i) + \langle w_1, x_i\rangle y_i + \\
&\quad \sum_{ij\in\mathcal{E}} \langle w_{00}, x_{ij}\rangle(1-y_{ij}) + \langle w_{11}, x_{ij}\rangle y_{ij}) \qquad (2)
\end{aligned}$$

Note that in the second step, we have eliminated the partition function, as it is not a function of a particular labeling. Also, $\mathcal{N}$ is the set of nodes in the MRF, $\mathcal{E}$ is the set of edges, $y_i \in \{0,1\}$ is the indicator that the ith node is labeled 1, and $y_{ij} = y_i y_j$. Features x_i are associated with the nodes, and features x_{ij} are associated with the arcs. Weights w_0, w_1, w_{00}, and w_{11} are associated with the nodes having labels 0 or 1, and both nodes having labels 00 or 11, respectively. The submodularity assumption requires that $\langle w_{00}, x_{ij}\rangle$ and $\langle w_{11}, x_{ij}\rangle$ be nonnegative. Note that we can enforce this by requiring nonnegative arc features and weights, which we will assume henceforth. Under these assumptions, following [10], this is a type of energy function that can be minimized via graph cuts.

To accomplish this, we first build a graph according to the construction procedure described in Kolmogorov. We first augment our original MRF graph with a special source (s) and a sink (t) node. For the unary potential associated with the node i, we add an arc $s \to i$ with weight $\langle w_0, x_i\rangle$, and an arc $i \to t$ with weight $\langle w_1, x_i\rangle$. For each binary potential, we add an arc $i \to j$ with weight $\langle (w_{00} + w_{11}), x_{ij}\rangle$; we also add weight $\langle w_{00}, x_{ij}\rangle$ to the arc $s \to i$ and add weight $\langle w_{11}, x_{ij}\rangle$ to the arc $j \to t$.

By the proof of Theorem 4.1 in [10], the minimizer of Eq. 2 corresponds to the minimum cut on the graph with this construction; nodes on the s side of the cut should be assigned a label of 0, and nodes on the t side should be assigned a label of 1.

B. Efficient learning

We now turn to the problem of estimating the weights w that induce a desired MAP labeling given features x. Our methodology in this section is based on the general framework presented by Taskar et al. [12] and parallels the development of Ratliff et. al [13], in which a very similar subgradient method is obtained for the problem of "maximum margin planning". The difference between their derivation and ours lies mainly in that ours specializes into a min cut problem, while theirs specializes into a shortest path problem.

We assume that we have constructed a graph with nodes $\mathcal{N}$ and edges $\mathcal{E}$ as described above, so that we can express the MAP estimate as a min cut. Following Taskar et al. [12], we formulate the learning problem as minimizing the norm of the weight vector subject to the constraint that the desired labeling "scores better" than an arbitrary labeling by an amount that scales with the Hamming distance between the desired and incorrect labelings. This leads to the following optimization in the case of separable data:

$$\begin{aligned}
&\min_{w\geq 0} && ||w||^2 \qquad (3) \\
&\text{subject to} && \min_{q\in\mathcal{Q}} \sum_{(ij)\in\mathcal{E}} \langle w, x_{ij}\rangle(q_{ij} - \hat{q}_{ij}) - \\
& && (N_n - \hat{q}_n^T q_n) \geq 0
\end{aligned}$$

Here we have formulated the problem in terms of min cuts. For simplicity of notation, we have introduced $\mathcal{Q}$ as the set of valid cuts. $q_{ij} \in \{0,1\}$ indicates whether arc $i \to j$ is cut; i.e., $q_{ij} = 1$ iff. node i and node j have different labels. q_n is a vector of cut variables for all terminal arcs.

These variables are of the form q_{si} and q_{it}; i.e., $q_{si} = 1$ implies that the label of node i is 0, and so forth. The cut corresponding to the desired labeling (henceforth called the desired cut) is designated $\hat{q}$. The difference between the cost of the min cut induced by the weights w and the desired labeling is therefore $\min_{q \in \mathcal{Q}} \sum_{(ij) \in \mathcal{E}} \langle w, x_{ij} \rangle (q_{ij} - \hat{q}_{ij})$. N_n is the number of nodes in the graph (excluding s and t). $N_n - \hat{q}_n^T q_n$ is thus equal to the number of disagreements between labeling q and the desired labeling.

We also assume that we have defined the weight vector as the concatenation of the individual weight vectors; i.e., $w = [w_0^T \; w_1^T \; w_{00}^T \; w_{11}^T]^T$. We therefore also assume that the feature vectors have been defined appropriately such as to induce the graph capacities described in section IV-A.

By rearranging terms, we obtain

$$\begin{aligned} \min_{w \geq 0} \quad & ||w||^2 & (4) \\ \text{subject to} \quad & \min_{q \in \mathcal{Q}} \sum_{ij \in \mathcal{E}} (w^T x_{ij} + \hat{q}_{ij}(\delta_{is} + \delta_{jt})) q_{ij} \\ & \geq N_n + \sum_{ij \in \mathcal{E}} (w^T x_{ij}) \hat{q}_{ij} \end{aligned}$$

where δ_{ij} is the Kronecker delta. This reveals a sort of stability nature to the constraint under adversarial tweaking of the problem data. If the constraint holds, we can increment by one the affinities of each node to the "wrong" terminal arc (the one that does not share its label), and the MAP labeling on the resulting graph will be the same.

This constraint has another important interpretation. The left-hand-side of the constraint is equivalent to augmenting the capacities of the original graph in a certain way and saying that the min cut on the new graph is equal to N_n plus the min cut of the old graph. Specifically, we are augmenting the capacities by adding one to the capacity of each cut terminal arc in the desired cut. Since each node has exactly one terminal arc cut in the desired cut, we are adding a total of N_n capacity to the graph.

Note that the left hand side of this constraint can never actually be strictly greater than the right hand side. This is because we cannot increase the max flow by more than N_n if we add a total of N_n capacity. In other words,

$$\begin{aligned} M(w) \quad := \quad & \min_{q \in \mathcal{Q}} \sum_{ij \in \mathcal{E}} (w^T x_{ij} + \hat{q}_{ij}(\delta_{is} + \delta_{jt})) q_{ij} \\ \leq \quad & N_n + \min_{q \in \mathcal{Q}} \sum_{ij \in \mathcal{E}} (w^T x_{ij}) q_{ij} \\ \leq \quad & N_n + \sum_{ij \in \mathcal{E}} (w^T x_{ij}) \hat{q}_{ij} & (5) \end{aligned}$$

Going back to the problem in (5), this allows us to replace the inequality with equality. We now also add a slack variable for the case where the constraint cannot be met, yielding:

$$\begin{aligned} \min_{w \geq 0} \quad & ||w||^2 + C\xi & (6) \\ \text{subject to} \quad & N_n + \sum_{ij \in \mathcal{E}} (w^T x_{ij}) \hat{q}_{ij} - \\ & \min_{q \in \mathcal{Q}} \sum_{ij \in \mathcal{E}} (w^T x_{ij} + \hat{q}_{ij}(\delta_{is} + \delta_{jt})) q_{ij} = \xi \end{aligned}$$

We now move the constraint into the objective to get

$$\begin{aligned} \min_{w \geq 0} ||w||^2 + C(N_n + \sum_{ij \in \mathcal{E}} (w^T x_{ij}) \hat{q}_{ij} - \\ \min_{q \in \mathcal{Q}} \sum_{ij \in \mathcal{E}} (w^T x_{ij} + \hat{q}_{ij}(\delta_{is} + \delta_{jt})) q_{ij}) \qquad (7) \end{aligned}$$

Though this might not look very helpful at first glance, it can be shown that this is a convex problem, so a global minimum exists. This formulation is also appealing in that the only constraints are simple nonnegativity constraints, which opens up the possibility of using a subgradient projection method [14].

To elaborate on this statement, we note that the objective is not differentiable everywhere. This is due to the presence of the min cut in the objective. Although there are some points where we will not be able to compute a gradient, if we can compute a subgradient in those places, we can still apply a subgradient-based optimization scheme.

To review, a subgradient of a function $f(w)$ is a vector q such that $f(\hat{w}) \geq f(w) + (\hat{w} - w)^T q$, $\forall \hat{w} \in \mathcal{R}^n$, if the dimension of w is taken to be n. A subgradient is a generalization of the gradient that may exist when the function is nondifferentiable. Unlike a gradient, a subgradient is not necessarily a descent direction, but following it does bring us closer to the optimum in a certain other sense. This fact induces a set of so-called subgradient optimization schemes for nondifferentiable convex problems that are analogous to gradient-based methods for differentiable problems.

To explore the possibility of using subgradient methods, we resort to Danskin's min-max theorem [14]. It can be shown via Danskin's theorem that at any w such that the min cut exists and is unique, the gradient of ξ exists and is equal the gradient of the term inside the minimization. At a w such that the min cut is degenerate, all valid min cuts are subgradients of ξ at w.

Define the matrix X such that the ith row of X is equal to the feature vector of the ith arc. Let $\tilde{q} = \arg\min_{q \in \mathcal{Q}} \sum_{ij \in \mathcal{E}} (w^T x_{ij} + \hat{q}_{ij}(\delta_{is} + \delta_{jt})) q_{ij}$ denote the vector of min cut variables given w. Denote by $\xi(w)$ the slack term in the objective of Eq. 7 as a function of w, and denote the subdifferential of $\xi(w)$ by $\partial \xi(w)$. By the chain rule and the discussion above,

$$(\hat{q} - \tilde{q})^T X \in \partial \xi(w) \qquad (8)$$

Therefore, given a "current" min cut vector associated with a current guess of w, a subgradient of the slack term is given by a signed sum of feature vectors corresponding to arcs where the current cut disagrees with the desired cut—if the arc is cut in the current cut but not the desired cut, it is subtracted; if the opposite is true, it is added. If we take a step in the direction of the negative cost subgradient, this is equivalent to adding capacity to arcs that were incorrectly cut and decreasing capacity of arcs that should have been cut, thus making bottlenecks at the desired cut, and eliminating bottlenecks at the undesired cut.

Therefore, performing a subgradient "descent" step on the slack term is very much like performing a weight vector update in the perceptron algorithm [15], in the sense that it makes an update to the weight vector proportional to the feature vector when there is a sort of disagreement between the desired and actual prediction.

To summarize, this analysis suggests a very simple subgradient projection method for minimizing the convex objective in (7). Given a current guess for w, find the min cut vector $\tilde{q}$ on the augmented-capacity graph to compute $\nabla_w \xi(w)$. Then take a step in the direction of a negative subgradient of (7), i.e., $-(2w + C(\hat{q} - \tilde{q})^T X)$, and project this onto the positive orthant. It is also interesting to note that this simple algorithm allows a trivial extension to regularization of w via other norms, although the L_2 norm may be well-motivated from the perspective of a convergence rate analysis [13].

In practice, this simple method often exhibits serious oscillations. Presumably this is due to the frequency of nondifferentiable points in the objective corresponding to sudden jumps in the cut vector; indeed, the cut vector can only change discontinuously. When this occurs, the calculated subgradient changes suddenly, and a step is taken back towards the last cut, which can cause a severe "chattering" phenomenon.

Fortunately, this shortcoming has a simple solution that works very well in practice—merely averaging the subgradient in time with an IIR filter usually helps convergence dramatically. This type of approach is referred to as an "averaged perceptron" method in the machine learning literature [16].

One issue we have ignored up to this point is how to handle a training set consisting of multiple disjoint graphs. This arises in our application, for example, as we wish to train on a set of disjoint graphs corresponding to each image. Although it would be possible to create a single large graph that happens to lack full connectivity, it can be seen by the properties of the max flow that one can equivalently calculate the contribution of each disjoint graph to the subgradient, and sum these. This is important in that (assuming we have N disjoint graphs) it allows us to perform N independent max flow computations, as opposed to a max flow computation on a graph N times larger than a single graph. Since the asymptotic performance of the best max flow algorithm is still superlinear in the problem data, this should provide a significant performance boost.

Finally, although we have presented a batch version of the training algorithm here, note that (like perceptron) it can just as easily be applied in an online fashion by changing the problem data after performing only a few subgradient steps.

V. Stereo for self-supervised learning

As mentioned in the introduction, we are primarily motivated by the problem of autonomous navigation. In this context it is often undesirable or impossible for a human to provide supervised input for a learning algorithm such as the one just described. For this reason we consider briefly in this section the problem of self-supervision with the aid of a stereo camera.

Our approach is based on a typical application of ground-plane stereo. Ground-plane stereo typically entails finding the largest planar region in the stereo disparity image. Since the ground around the robot is assumed to be locally flat, it is assumed that this plane corresponds locally to the ground. Any points that are outliers with respect to the current ground plane model are assumed to be obstacles.

Concretely, let x_w be a point on the ground plane in Euclidean coordinates, referenced to an arbitrary frame of reference. A plane in this space can be parametrized by a vector c such that $c_w^T x_w = 1$. It can be shown that under the projective transformation of a pinhole camera model, every such ground plane corresponds to a linear model in image coordinates and disparity; i.e.,

$$\alpha_0 x + \alpha_1 y + \alpha_2 = d \tag{9}$$

where (x, y) are the image coordinates of a pixel with observed stereo disparity d. Given observations of (x, y, z), we can treat this as a linear least squares problem where we attempt to predict d as an affine function of (x, y) with parameters α. This corresponds to optimizing the following objective:

$$\min_{\alpha} \sum_{i=1}^{N} ||\alpha_0 x_i + \alpha_1 y_i + \alpha_2 - d_i||_2^2 \tag{10}$$

Here N observations are indexed by i. Although this yields a simple linear least squares solution, there is a problem with this objective in that it heavily penalizes outliers, which means that our ground plane estimate would be heavily affected by points not on the ground plane.

With this in mind, we employ a robust least squares procedure to estimate the ground plane. This procedure entails performing successive iterations of weighted linear least squares regression. In each step, points contributing to the ground plane estimate are assigned weights w_i by a certain function designed to down-weight outliers from the current estimate. For instance, this function could be similar to a Gaussian:

$$w_i \propto e^{-(\alpha_0 x_i + \alpha_1 y_i + \alpha_2 - d_i)^2 / (2\sigma^2)} \tag{11}$$

A weighted least squares model is then fit to the data using these observation weights; this is equivalent to linear least squares after multiplying the ith observation by $\sqrt{w_i}$. This procedure is iterated to convergence, and usually results in an robust estimate of the ground plane. Although we have presented only heuristic arguments for the correctness of this procedure here, we note that a very similar procedure was shown in [17] to optimize a robust L_1-norm objective.

This procedure has a number of properties that make it well-suited to online applications; namely, it is naturally an iterative method that can be used to slowly refine a ground plane estimate in an online way. This gives it a natural sort of regularization effect that is not evident in other popular methods, such as RANSAC [18].

Once we have a ground plane estimated via this method, we use it to classify each pixel for which there is valid disparity as either an obstacle or ground. This can be accomplished by a simple threshold on the residual of the fit for each observation; i.e., the ith pixel is labeled ground if $|\alpha_0 x_i + \alpha_1 y_i + \alpha_2 - d_i| \leq d_g$ (where d_g is a constant threshold), and is labeled obstacle if the residual exceeds another threshold. This short-range classification is then used as input to the learning algorithm.

VI. Experimental results

We performed a number of experiments to evaluate the method's classification accuracy and computational efficiency. Our experimental platform consisted of the DARPA Learning Applied to Ground Robots (LAGR) standard platform. The LAGR robot features two Bumblebee stereo cameras, each with a 12 cm baseline. For the purposes of these experiments, the LAGR robot was used to collect data that was logged and then processed offline. We implemented the algorithms in MATLAB, with the exception of the graph cut implementation, for which we used Y. Boykov and V. Kolmogorov's publicly available code [19].

A. Implementation details

1) Feature selection: For simplicity, we chose to use only RGB color features for our experiments. Although these are not the ideal features for our purpose, we note that better features would only improve the performance of our method, making the data more separable. In any case, some care must be taken in order to ensure that a linear classifier can separate data well with the desired features. Keeping this in mind, we quantized RGB space into 512 bins. Our node features (x_i) then consisted of indicator vectors indicating the bin occupied by the example RGB value. In other words, if an RGB value falls in bin n, the corresponding feature vector consists of the vector of all zeros except for a single 1 in element n. This set of indicator features was chosen because a linear classifier on these features can produce any arbitrary assignment of labels to the bins. Arc features were constructed by concatenating the node features of the head of each arc with the features of its tail. We note again that this form of quantization and creation of arc features could be used for any arbitrary set of features.

2) Graph construction: For each image in the training set, we constructed our MRF by associating a node with each pixel for which valid stereo data was available. Arcs were added between each pixel and its four "north-south-east-west" neighbors, minus those without valid stereo information. Each arc was associated with the features described above.

	% Correct Histogram	% Correct MRF	% Correct MRF - Histogram
"Easy" training set	88.40	94.85	6.46
"Hard" training set	72.47	84.99	12.52
"Easy" held-out set	83.50	92.21	8.71
"Hard" held-out set	68.26	79.76	11.50

TABLE I

Performance of histogram and MRF classifiers in batch training experiment

B. Batch training

For the purpose of comparison, we also implemented a histogram-based classifier that classifies each pixel independently. The histogram classifier is very simple. We first quantize RGB space into the same 512-bin partition used for the MRF classifier. Each bin then counts the number of times its color was observed as an obstacle and the number of times its color was observed as ground. The bin is then classified as ground if the ground count exceeds the obstacle count, and vice-versa. For bins with the same counts, the label was randomly assigned with a probability proportional to the relative frequency of ground pixels versus obstacle pixels. This was done to avoid bias in the classifier.

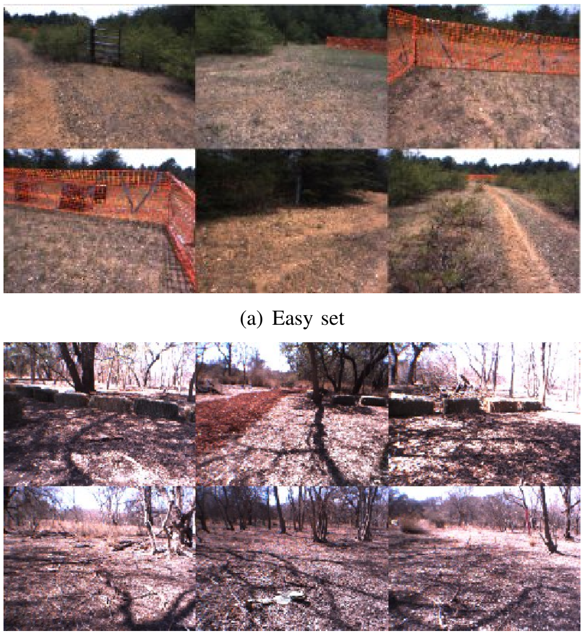
The histogram is particularly nice for comparison because it can be thought of as acting on exactly the same set of indicator features as we use in the MRF for node features, since the histogram is simply the sum of all the indicator features. Also, though it seems simple, note that the histogram classifier always yields optimal performance on the test set given these indicator features, if we measure performance in terms of the proportion of correctly labeled examples.

In the batch training experiment, we first acquired 100 sequential images from a LAGR robot log. We then trained the MRF and histogram on 10% of the images and evaluated them on the remaining 90%. We manually tuned the slack penalty parameter C until a good balance was achieved between complexity and test set accuracy.

We performed the experiment on two independent datasets. Sample images from these datasets are shown in Fig. 1. The first dataset is considered much easier than the other, as it features much better color contrast and exposure than the other set.

Table I shows the results of the experiment in terms of correct classification rate. The results show that the MRF performs significantly better in both discriminative power and generalization ability on both datasets, in the sense that classification rates greatly improved on both the training and test sets. The difference is especially pronounced in the hard dataset, where we saw an 11.5% increase in classification accuracy on the held-out set over the independent classifier.

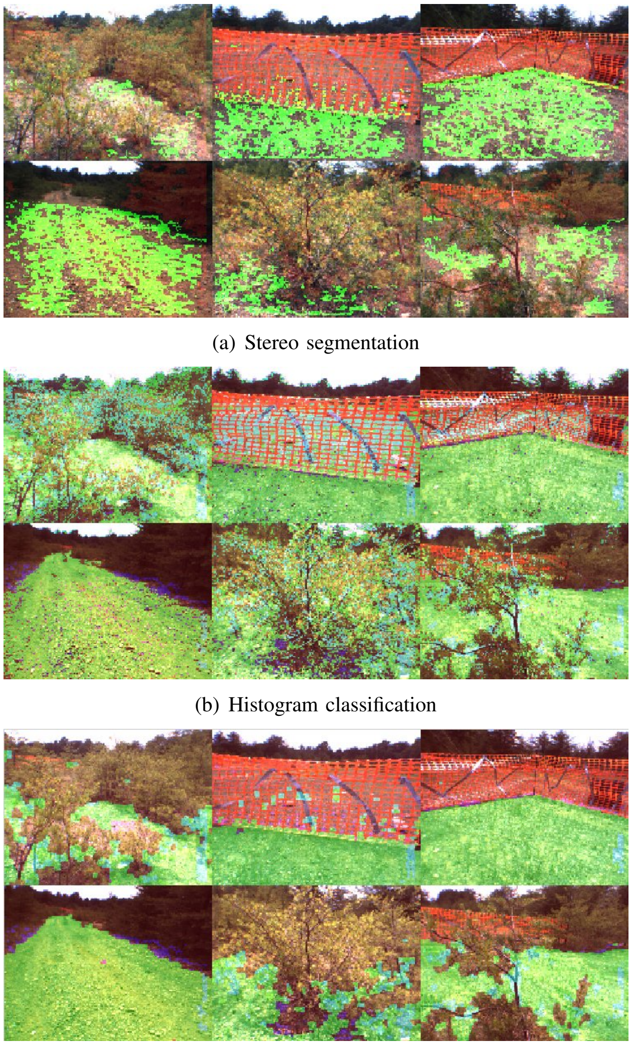
Fig. 2 shows some sample classified images based on stereo, the MRF, and the histogram classifiers. It is clear here that the MRF is able to make large gains by uniformly and correctly classifying large image patches that are classified in a very inconsistent way by the histogram.



(a) Easy set

(b) Hard set

Fig. 1. A small sample of images used in the experiments

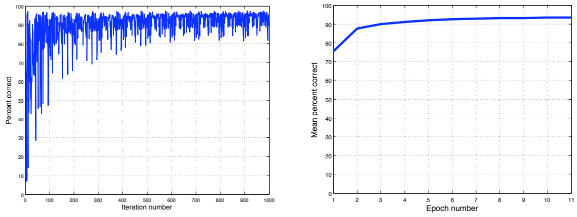


(a) Stereo segmentation

(b) Histogram classification

(c) MRF classification

Fig. 2. Sample stereo segmentation results and classification results via MRF and histogram. Green-tinted pixels are labeled ground, and red pixels are labeled obstacles. In the MRF and histogram classifications, cyan pixels are mislabeled as ground, and dark blue pixels are mislabeled as obstacles.



(a) Per iteration

(b) Per epoch

Fig. 3. Plot illustrating online classification performance on a randomly permuted set of 100 images. Per iteration performance shows performance of classifier just before training on the nth image. Per epoch performance shows average percent correct over all images in each epoch.

C. Online training

We also investigated the scenario of online training, since this is the scenario that primarily motivates our work. As mentioned before, the adaptation to the online scenario is straightforward and consists merely of performing a gradient step per image.

In practice, however, we found one slight modification to be very useful in the online scenario. Instead of using $||w||^2$ as a regularizer, we use $||w - w_i||^2$, where w_i is the weight vector at the ith iteration (or image). This change, inspired by [20], encourages very conservative changes to the weight vector. We found this to work much better in the online scenario than the simple norm regularizer, which seems too harsh, causing the classifier to “forget” too easily.

To evaluate the classifier’s performance in this scenario, we took the same “easy” dataset used before and randomly permuted it. We then trained the classifier by taking a single subgradient step on each image’s data in succession. We repeated this procedure, looping through the entire dataset multiple times. At each step, we evaluated the classifier on the current image before training it on the image. Fig. 3 shows these results.

These results show that even when trained sequentially in a random order, the classifier seems to monotonically improve its performance in each epoch of training. By the 11th epoch, the mean classification accuracy per image had reached 93.6%.

D. Computational efficiency

A critical aspect of the method described here is its computational efficiency, as our intent is to run the method on a robot in real-time. We have found that even unoptimized MATLAB code is sufficient for real-time performance. The

	Subgradient calculation (ms.)	MAP inference (ms.)
Mean	37.2	7.6
Standard deviation	5.1	2.2

TABLE II

Performance statistics for online classifier in milliseconds per 160x120 image (N.B.: subgradient calculation time includes MAP inference time).

only exception in our current implementation is the feature calculation steps, which a proper implementation should handle in negligible time due to the simplicity of the features we have used. We therefore discount the feature calculation time in the following.

We quantified the computational efficiency of the method by performing timings on a machine equipped with a 2.4 GHz Intel Xeon CPU. Performance statistics for 160x120 images are summarized in Table II. An entire subgradient calculation step, which includes the MAP inference step as a subroutine, takes less than 40 milliseconds, which is more than suitable for real-time operation, assuming we take one subgradient step per image. We note that, counter-intuitively, the overhead of the subgradient calculation actually far surpasses the time spent in the graph cut routine. We attribute this to an artifact of our implementation; in an optimized implementation, the subgradient calculation overhead should probably be less than the time spent in the graph cut. Although performance statistics were unavailable as of the time of this writing for larger images, we note that performance on 320x240 images was also well within the realm of real-time performance, chiefly due to the excellent scaling properties of the graph cut inference routine.

VII. Conclusions

We have shown how it is feasible to apply discriminatively trained submodular MRFs to the problem of online visual learning for an autonomous robot. MAP inference is performed by an efficient max flow computation, and learning proceeds via a an averaged subgradient method that resembles the averaged perceptron method for max margin classification.

Our experiments have shown that the MRF significantly outperforms independent classification with the same features on multiple benchmarks and datasets. We have also seen empirically how the MRF can be trained in either an online or batch method to yield high accuracy rates, making the method very flexible with regard to different learning scenarios.

Although results are very promising with the current methods and implementation, we still hope to achieve even better performance by pursuing a number of avenues that may provide large gains. We have used only very small neighborhoods in the current implementation; in the future, we hope to study the effect of increasing neighborhood size, perhaps via the use of a pyramidal graph structure. It is also expected that more interesting node and edge features will yield much improved performance. Finally, we hope to improve the computational efficiency of the method even further by warm-starting the max flow computation with a previous solution. This should significantly boost performance of both the learning and inference steps, as we assume the problem data would be slowly changing in time.

Acknowledgments

This work was supported in part by the DARPA LAGR program.

References

[1] S. Thrun, M. Montemerlo, H. Dahlkamp, and D. Stavens, “Stanley: The robot that won the DARPA Grand Challenge,” *Journal of Field Robotics*, 2006.

[2] B. Sofman, E. Lin, J. A. Bagnell, J. Cole, N. Vandapel, and A. Stentz, “Improving robot navigation through self-supervised online learning,” *Journal of Field Robotics*, 2006.

[3] I. Ulrich and I. Nourbakhsh, “Appearance-based obstacle detection with monocular color vision,” in *Proceedings of the AAAI National Conference on Artificial Intelligence*, 2000.

[4] J. Sun, T. Mehta, D. Wooden, M. Powers, J. Rehg, T. Balch, and M. Egerstedt, “Learning from examples in unstructured, outdoor environments,” *Journal of Field Robotics*, 2006.

[5] A. Howard, M. Turmon, L. Matthies, B. Tang, A. Angelova, and E. Mjolsness, “Towards learned traversability for robot navigation: from underfoot to the far field,” *Journal of Field Robotics*, 2006.

[6] C. Wellington, A. Courville, and A. Stentz, “Interacting Markov random fields for simultaneous terrain modeling and obstacle detection,” in *Proceedings of Robotics: Science and Systems I*, 2005.

[7] A. Torralba, K. P. Murphy, and W. T. Freeman, “Contextual models for object detection using boosted random fields,” in *Neural Information Processing Systems*, 2004.

[8] X. He, R. S. Zemel, and M. Á. Carreira-Perpiñán, “Multiscale conditional random fields for image labeling,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2004.

[9] C. A. Bouman and M. Shapiro, “A multiscale random field model for Bayesian image segmentation,” *IEEE Transactions on Image Processing*, March 1994.

[10] V. Kolmogorov, “What energy functions can be minimized via graph cuts?” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Februrary 2004.

[11] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.

[12] B. Taskar, V. Chatalbashev, and D. Koller, “Learning associative Markov networks,” in *Proceedings of the International Conference on Machine Learning*, 2004.

[13] N. D. Ratliff, J. A. Bagnell, and M. A. Zinkevich, “Maximum margin planning,” in *Proceedings of the 23rd International Conference on Machine Learning*, 2006.

[14] D. P. Bertsekas, *Nonlinear Programming*. Athena Scientific, 2003.

[15] F. Rosenblatt, “The perceptron: a probabilistic model for information storage and organization in the brain,” *Psychological Review*, 1958.

[16] M. Collins, “Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms,” in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, July 2002.

[17] Y. Lin and D. D. Lee, “Bayesian L$_1$-norm sparse learning,” in *International Conference on Acoustics, Speech, and Signal Processing*, 2006.

[18] M. A. Fischler and R. C. Bolles, “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography,” *Communications of the ACM*, 1981.

[19] Y. Boykov and V. Kolmogorov, “An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, September 2004.

[20] K. Crammer and Y. Singer, “Ultraconservative online algorithms for multiclass problems,” *The Journal of Machine Learning Research*, March 2003.