Prior Data and Kernel Conditional Random Fields for Obstacle Detection

Carlos Vallespi
National Robotics Engineering Center
Carnegie Mellon University
Pittsburgh, Pennsylvania 15201
Email: cvalles@ri.cmu.edu

Anthony (Tony) Stentz
National Robotics Engineering Center
Carnegie Mellon University
Pittsburgh, Pennsylvania 15201
Email: axs@ri.cmu.edu

***Abstract*— We consider the task of training an obstacle detection (OD) system based on a monocular color camera using minimal supervision. We train it to match the performance of a system that uses a laser rangefinder to estimate the presence of obstacles by size and shape. However, the lack of range data in the image cannot be compensated by the extraction of local features alone. Thus, we investigate contextual techniques based on Conditional Random Fields (CRFs) that can exploit the global context of the image, and we compare them to a conventional learning approach. Furthermore, we describe a procedure for introducing prior data in the OD system to increase its performance in "familiar" terrains. Finally, we perform experiments using sequences of images taken from a vehicle for autonomous vehicle navigation applications.**

I. Introduction

Obstacle detection (OD) is important in many mobile robot applications and autonomous vehicles. The most successful OD systems rely on range information to detect obstacles by size and shape. Among all the range sensors, laser rangefinders are the most popular and widely used range sensors, due to their quality of data.

Unfortunately, laser rangefinders contain mobile parts in their design which makes them complex and expensive. Furthermore, they may require extra hardware to scan the scene and a precise calibration. In contrast, color cameras are mass-produced and are comparatively inexpensive. However, the lack of range data makes the OD problem more challenging. Local features alone are not enough to extract enough information to detect obstacles reliably. We alleviate this problem by exploiting the contextual information in the image. For example, since we cannot measure the shape of the rock directly, we learn that rocks are gray objects with certain texture properties and surrounded by brown dirt.

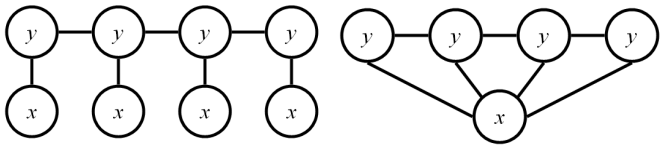
This paper investigates several contextual techniques based on Conditional Random Fields (CRFs), which have been successfully used in the past for classification and segmentation tasks, and allow to exploit contextual features in the image. One of the CRF models presented in this paper uses a log-linear model which imposes a linear combination of the input features. Then, we apply the "kernel trick" to this model to allow the use of different kernels with the hope that, in the projected space, classes become linearly separable.

1: A John Deere tractor (4710 series) was used for some of the experiments in agricultural applications. It was equipped with cameras, positioning sensors, a computer and a SICK laser for perception.

Moreover, we present an algorithm to improve the OD system performance in agricultural environments and, in general, in applications where the system re-visits the same areas. In this case, we build a database of hand -or automatically- classified images of the terrain and integrate them in a contextual model to improve the obstacle detection process.

The contextual methods presented in this paper are compared to conventional non-contextual learning approaches. We use a OD system equipped with a SICK laser, a color camera, positioning sensors (IMU, GPS and wheel encoders for speed measurements) to extract the precise location of the obstacles (see fig. 1). Then, all the algorithms are trained to match the performance of this OD system.

This paper is structured as follows. The next section gives a short overview of prior work in this area. In section III we introduce the basic CRF model. Section IV describes our procedure to apply the kernel trick to a log-linear CRF model. In section V we show an algorithm to make use of prior data.



2: **Left:** MRF graph structure. **Right:** CRF graph structure.

Finally, we show the effectiveness of our algorithm in a batch of experiments in section VII.

II. Related Work

In the obstacle detection and avoidance field, Dima presented an algorithm which combined the information provided by different sensors in the image frame [3]. The data from each sensor was transformed to image coordinates, thus each part of the image contained features from the image plus features from other sensors (i.e. infrared, LADAR, etc). Whereas the algorithm enabled the fusion of heterogeneous sensors, the classifier that was used assumed independence among all the parts in the image.

Many cues needed for obstacle detection are contextual information, and Markov Random Fields (MRFs) are widely used machine learning tools to exploit this information. However, in the MRF framework, the observed data is assumed to be conditionally independent which can be very restrictive in some applications (see fig. 2). Unlike MRFs, CRFs model directly the conditional distribution. Thus, the relations between the input variables do not need to be explicitly represented. In the past, their main limitation was the use of slow training algorithm (such as iterative scaling -IIS-); however, recent advances in CRF theory have found efficient algorithms for parameter learning and inference in general CRF graphs [15].

Log-linear CRFs have been successfully used in the past for image labeling. In this form, CRFs allow for a parameter estimation guaranteed to find the global optimum due to the convex property of their conditional likelihood function. For instance, CRFs have been used for detection of man-made structures in natural images [7]. CRFs have been used for object detection and recognition given its parts in images [11]. They have been also used for object segmentation tasks in images [14], and with occlusion handling [17]. Also, this idea has been extended to segmentation in video sequences [16]. Saxena el al. [12] uses a discriminative MRF model to estimate the depth using a single still camera, which could be used as the input to an obstacle detection algorithm.

A linear relation between features and random variables in the CRF model has been widely used, but in some cases this can be a restrictive constraint. To overcome this limitation, an extension to the CRF model to allow the use of custom kernels was proposed by [8]. In this paper, we present an alternative algorithm. In general, the possibility of changing the kernel allows the model to adapt better to a specific problem resulting in better performance.

III. Conditional Random Field model

A. CRF model

A Conditional Random Field (CRF) is an undirected graphical model in which edges represent conditional dependencies between random variables at the nodes. The distribution of each random variable y_i is conditioned on an input sequence $\mathbf{x}$. The conditional dependency of the random variables on $\mathbf{x}$ is defined by using feature functions with some associated weights. Together, they can be used to determine the probability of each y_i. Dependencies among the input variables $\mathbf{x}$ do not need to be represented because the model is conditional, affording the use of complex and rich features of the input. Thus, CRFs are discriminative models, that is, they model $p(\mathbf{y}|\mathbf{x})$[1].

In a general way, to model the conditional probability distribution of a sequence of labels $\mathbf{y}$ given the observations $\mathbf{x}$, $p(\mathbf{y}|\mathbf{x})$ takes the form shown in (1):

$$p(\mathbf{y}|\mathbf{x}) = \frac{1}{\mathbf{Z}(\mathbf{x})} \prod_{c \in \mathbf{C}} \Psi_c(\upsilon_{\mathbf{c}}), \tag{1}$$

where $\Psi_c(\upsilon_{\mathbf{c}})$ is a potential function that depends on the variables in a cluster c (defined as υ_c). $\mathbf{Z}(\mathbf{x})$ is called the partition function, and it is a normalization factor to make sure that $\sum_{y_i} p(y_i|\mathbf{x}) = 1$. It depends on the data, therefore it takes different values as the input ($\mathbf{x}$) changes.

In this paper, we use a log-linear model for the CRF. Thus, Ψ are potentials of the form shown in (2):

$$\Psi_c(\upsilon_{\mathbf{c}}) = \exp\left\{ \sum_{\mathbf{k}} \lambda_{\mathbf{k}} f_k(\mathbf{x_c}, \mathbf{y_c}) \right\}, \tag{2}$$

where $\mathbf{x_c}, \mathbf{y_c} \in \upsilon_{\mathbf{c}}$ and f_k is feature k function over $\mathbf{x}, \mathbf{y}$.

In the image labeling task, the CRF model we use is a lattice, forming an undirected graph $G = (\mathbf{V}, \mathbf{E})$. $\mathbf{V}$ are the nodes or vertices and $\mathbf{E}$ are the edges. Every node and every edge contain a potential function that operates on a subset of the random variables present in G. Thus, we define the conditional probability distribution of the CRF as shown in (3):

$$\begin{array}{c} p(\mathbf{y}|\mathbf{x}) = \frac{1}{\mathbf{Z}(\mathbf{x})} \prod_{i \in \mathbf{V}} \Psi(x_i, y_i) \prod_{(i,j) \in \mathbf{E}} \Psi(y_i, y_j, x_i, x_j) \\ Z(\mathbf{x}) = \sum_y \prod_{i \in \mathbf{V}} \Psi(x_i, y_i) \prod_{(i,j) \in \mathbf{E}} \Psi(y_i, y_j, x_i, x_j), \end{array} \tag{3}$$

where, functions Ψ are of the form:

$$\begin{array}{c} \Psi(x_i, y_i) = \exp\left\{ \sum_k \mu_{k y_i} g_k(x_i) \right\} \\ \Psi(y_i, y_j, x_i, x_j) = \exp\left\{ \sum_k \lambda_{k y_i y_j} f_k(x_i, x_j) \right\} \end{array} \tag{4}$$

We set $\mu \in \Re^{K \times L}$ and $\lambda \in \Re^{K \times L \times L}$. L is the number of different labels or classes and K is the number of features.

[1]Bold letters denote an array of elements or variables. Functions are represented by non-bold letters followed by parentheses, i.e. $p(\mathbf{x})$. All non-bold no-function letters represent variables. In the CRF context, unless stated differently, $\mathbf{x}$ denotes data and $\mathbf{y}$ denotes labels. Sub-indexes denote elements of the array, i.e. x_i denotes the data at the i-th node. y_i denotes the label at the i-th node. $\{y_i, y_j\}$ denotes a pair of labels of $\mathbf{y}$ at nodes i, j. $\mathbf{y} = \{y_i, y_j\}$ represents all pair of labels of adjacent nodes of $\mathbf{y}$ equal to the pair $\{y_i, y_j\}$. $\mathbf{y}^m$ denotes the m-th sequence of $\mathbf{y}$ labels.

Unlike other representations found in the literature, we chose functions g_k and f_k which only depend on the data, and not on the labels. Because of this representation, the weights are the ones that depend on the labels. Thus, to account for the different classes, $L-1$ hyper-planes are needed.

The total number of node weights is $(L-1)K$ and it is equivalent to use $L \times K$ node weights and set $\forall k = 1..K \;\; \mu_{kL} = 0$. We chose the edge weights to be the absolute value of the difference of features in adjacent nodes. Hence, the total number of edge weights in this representation is $L \times L \times K$ because we need as many weights as node features and combinations of pairs of labels. However, we restrict $\lambda_{kls} = -\lambda_{kll} \forall l \neq s$, which reduces the number of edge weights to $L \times K$.

It is interesting to note that the model defined in (3) contains a logistic regression classifier in each node. Simply, by setting the edge weights to 0 (i.e., define $\forall i,j \in \mathbf{E} \;\; \lambda_{ky_iy_j} = 0$) it is easy to see that every node contains a multi-class logistic regression classifier.

In summary, the set of parameters for the CRF in our representation is the union of the node weights and the edge weights ($\phi = \{\mu_{1...K,1...L-1}, \lambda_{1...K,1...L}\}$), giving a total of $K \times (2L-1)$ parameters.

B. Inference in 2D CRF

It is worth noting that inference problems like marginalization and maximization are NP-hard to solve exactly and approximately (at least for relative error) in lattice graphical models, and in general, for most of the graph structures. In a CRF graph model, maximization is to find the most likely sequence of labels $\mathbf{y}$ given an input $\mathbf{x}$, that is:

$$\mathbf{y}^{max} = \underset{\mathbf{y}}{\text{argmax}} \; p(\mathbf{y}|\mathbf{x}, \phi) \tag{5}$$

However, finding $\mathbf{y}^{max}$ exactly is infeasible in practical cases for 2D CRFs. A brute force algorithm would need to explore all possible labelings, which in a binary CRF of size 24×32 would be 1.5×10^{231}. In theory, the marginalization problem for graphical models with loops is #P-complete and maximization is NP-complete. Thus, approximate inference is used to solve these problems.

There are several methods in the literature for approximate inference in graphs (i.e. maxent -although only works for binary labels-, variational methods, Monte Carlo methods, Belief Propagation (BP), etc...). We used BP for approximate inference, because it gives good results in practice [9] and provides solution to the marginalization and maximization problems.

C. Maximum Likelihood parameter learning

In our work, we use the MLE principle to learn the parameters ϕ such that the regularized negative log-likelihood is minimized. The algorithm assumes that we are given set of i.i.d. labeled images $(\mathbf{X}^m, \mathbf{Y}^m) \in \mathbf{M}$. Regularization is added in the form of a Gaussian centered at 0 over the parameters to avoid over-fitting. The negative regularized log-likelihood for a CRF model is given by (6):

$$nll(\phi) = -\sum_{m \in \mathbf{M}} \log\{p(\mathbf{y}|\mathbf{x}^m, \phi)\} + \frac{\lambda}{2}\phi^T\phi \tag{6}$$

Ignoring the regularization term, the derivatives of the log-likelihood over the parameters ϕ yield to equations in (7). The first term is the value of the features under the empirical distribution. The second term, which arises from the derivative of $\log Z(\mathbf{x})$, is the expectation of the features under the model distribution. Equation (7) shows the derivatives corresponding to μ and λ, respectively:

$$\begin{aligned} \frac{\partial nll(\phi)}{\partial \mu_{kl}} &= -E_{\tilde{p}(y=l,\mathbf{x})}[g_k] + E_{p(y=l|\mathbf{x};\phi)\tilde{p}(\mathbf{x})}[g_k] \\ \frac{\partial nll(\phi)}{\partial \lambda_{kls}} &= -E_{\tilde{p}(\mathbf{y}=\{l,s\},x)}[f_k] + E_{p(\mathbf{y}=\{l,s\})|\mathbf{x};\phi)\tilde{p}(\mathbf{x})}[f_k] \end{aligned} \tag{7}$$

Unfortunately, there is no analytic solution to this equation (setting the gradient to 0 and solving for λ does not always yield to a closed-form solution). Thus, an iterative algorithm is needed in order to approximate the optimal solution. Note that the function $nll(\phi)$ is concave, which follows from the convexity of functions of the form $g(\mathbf{x}) = \log \sum_i \exp x_i$. This property shows that every local optimum is also a global optimum. Adding *L2* regularization to the NLL ensures that $\log l(\phi)$ is strictly concave, which implies that it has exactly one global optimum.

Thanks to this property, methods like steepest descent can be used, although they may require many iterations to converge making them slow. Newton's method can converge much faster because it takes into account the curvature of the likelihood. However, computing the Hessian can be expensive, too, since it is quadratic in the size of the parameters. An intermediate solution for this problem is to use quasi-Newton methods, such as the Broyden-Fletcher-Goldfarb-Shanno (BFGS) method [1, 4, 6, 13], in which the Hessian is updated by analyzing successive gradient vectors. BFGS is the algorithm that we use in this work for optimization (through `fminunc` in Matlab), and, as it is discussed in [15], it provides a rapid convergence to the optimal solution.

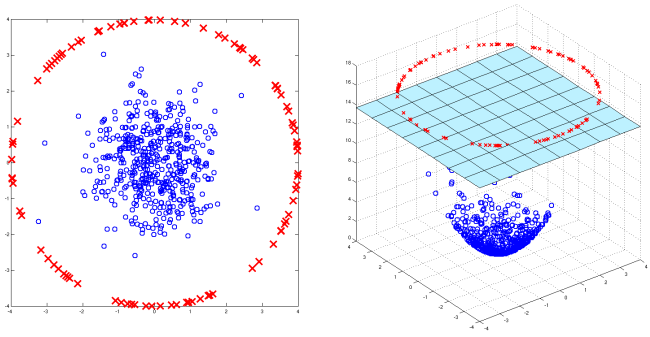
IV. Kernel Conditional Random Fields

A. "Kernel Trick" and Logistic Regression

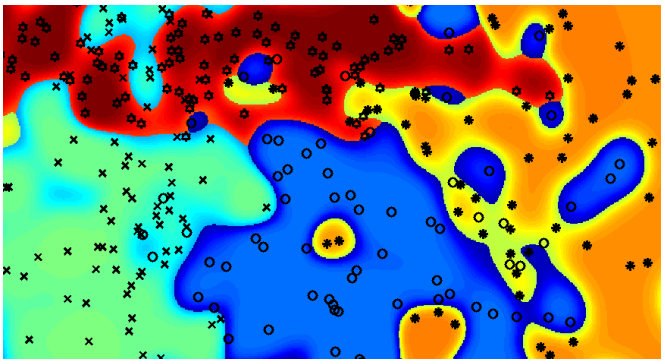
Logistic Regression (LR) is a discriminative linear classifier that estimates the $p(y|x)$ by using a linear combination of features of x. The conditional likelihood for LR is defined in (8):

$$p(y|x) = \frac{\exp\{\sum_k \mu_{ky} g_k(x)\}}{1 + \sum_{y=1}^{y=L-1} \exp\{\sum_k \mu_{ky} g_k(x)\}} \tag{8}$$

By applying the "kernel trick", one can convert a linear classifier algorithm into a non-linear one by using a non-linear function to map the original observations into a higher-dimensional space; this makes a linear classification in the new space equivalent to non-linear classification in the original space (see fig. 3). We apply the Mercer's theorem, which states that any continuous, symmetric, positive semi-definite kernel



3: Example of a dataset not linearly separable in the original dimension space (on the **left**), but it becomes separable after using a quadratic kernel (on the **right**), augmenting the dimensionality of the input space by 1.



4: Example of boundaries obtained by a multi-class IVM classifier, 36 import vectors and a a Gaussian kernel ($\sigma = 0.1$).

function $K(x_i, x_j)$ can be expressed as a dot product in a high-dimensional space, to equation 8:

$$p(y|x) = \frac{\exp\left\{\sum_{x_i \in S} \alpha_{iy} K(x, x_i)\right\}}{1 + \sum_{y=1}^{L-1} \exp\left\{\sum_{x_i \in S} \alpha_{iy} K(x, x_i)\right\}}, \quad (9)$$

where S is the space of vectors that span the kernel. As with LR, the log-likelihood is a convex function, and it is possible to compute the gradient and the Hessian of the log-likelihood, making suitable Newton-Raphson methods for rapid optimization. However, there are several performance penalties by using this method:

- Computing the kernel matrix can be computationally expensive ($O(N^2)$, where N is the number of vectors).
- A $N \times N$ matrix must be inverted at each iteration of Newton-Raphson method, increasing the computational cost of the training algorithm to the order of $O(N^3)$.
- In practice, most (if not all) α_i have non-zero values, which increases the cost of classifying samples (each new sample needs to be projected into all the x_i in the kernel).

These problems can be solved by fixing S to use a small subset of x_i. However, we need an algorithm to determine which and how many x_i should be in S.

In this paper, we will use the method proposed by [18]. The sub-model S found by IVM algorithm is an approximation to the full model found by KLR. The algorithm starts with an empty set of vectors for S. Then, at every iteration the vector that minimizes de NLL the most is added to S. The vectors in the kernel space S are called import vectors. The algorithm stops when the NLL does not decrease after some number of iterations. A toy example is shown in fig. 4.

B. "Kernel trick" for CRFs (K-CRFs)

K-CRFs were originally introduced in [8], and they allow the use of implicit feature spaces through Mercer kernels. Our approach differs mainly in the way we find the kernel space. In our algorithm, we use the IVM algorithm as described in section IV-A to find the vectors x_i that will span the kernel space S in the node potentials. Then, we compute the g_k projected into the kernel space found by the IVM algorithm in order to extract the set of node features that will be used by KCRF. Finally, we use the algorithm described in section III-C for parameter learning till its convergence.

In this work, we experiment with Gaussian kernels because they often provide good performance [2]. When they are used, the corresponding feature space is a Hilbert space of infinite dimension. However, the regularization used for parameter learning avoids the infinite dimension to spoil the results. In this paper, we refer to the Gaussian kernels the ones that take the form in (10):

$$K(x_i, x_j) = e^{-\frac{\|x_i - x_j\|^2}{2\sigma^2}}, \quad (10)$$

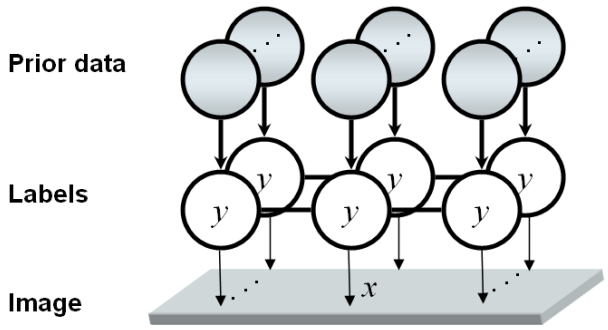
where x_i, x_j are input vectors.

V. Prior data and K-CRFs (PK-CRF)

In this section, we introduce an algorithm to take advantage of situations where prior labeled data is available (i.e., agricultural applications where vehicles revisit the same areas multiple times). If we label the data that corresponds to the working area once, we may be able to use these labelings to improve future labeling performances. Thus, in cases where the input image is similar to one in the prior data set, we may be able to re-use the prior labels. The parts of the image which experience changes (i.e. illumination, pose, new/missing objects, etc) may need new evaluations.

The algorithm described in this section assumes that the input data is tagged with pose (which does not need to be exact), and that the ground plane is mostly flat. For this task, we build a database with the prior labeled images tagged with pose. Then, for every new input image, the closest image in pose is recovered from the database and aligned with the input image (see section VI). At this point, we can refer each part/region in one image to the other. Thus, if we know the ground truth for the labels of one of the images, we can use this information in the image that we are trying to label.

In practice, there are differences between the input image and the reference image, even after the alignment, due to errors in the alignment process, errors in pose, possible changes in the environment, moving obstacles, etc. However, we let the CRF to decide for us whether the label in the reference image should be used.



5: PK-CRF model. White nodes represent random variables.

We added the prior data into the CRF model in the form of binary features for the nodes and real features for the edge. These features incorporate labeling information of the reference image to the input image. The node features have a value of 1 if the hypothesis about the label in the input image is the same as the reference image, and 0 otherwise. The edge features contain information about the differences between the region in the input image and the region in the reference image (similar to the edge features in the CRF). We refer to these features as h_k.

Each of these features is multiplied by a weight that depends on the class ($\gamma_n l$). Putting together this new features with the CRF model, we get the following node potentials:

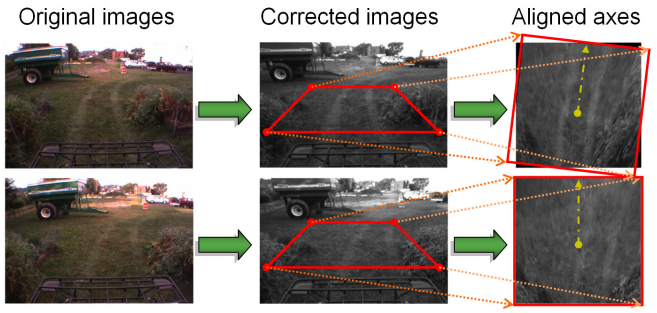
$$\Psi(x_i, y_i) = \exp\left\{\sum_k^K \mu_{ky_i} g_k(x_i) + \sum_n^N \gamma_{ny_i} h_n(x_i)\right\} \quad (11)$$

Where, K is the number of node features and N is the number of prior features for each node. Fig. 5 shows a graphical representation of the model. Discrete random variables are connected to the image features and also connected to the features and label information from the reference image. Some of the prior features may be missing in the cases where no match is found for a patch in the reference image.

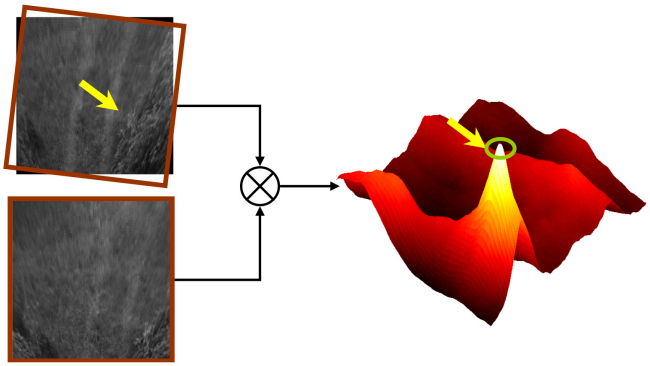
By adding these features we expect the CRF model to learn that if the node of the image and the node in the reference image are similar, then there should be a bias towards using the same label in the input image as the one used in the reference image. However, if the regions differ, then the information coming from the image taken in the past should be discarded and a full evaluation of the region would be required. We called this model Prior K-CRF (PK-CRF).

VI. Image Registration

We use an algorithm for aligning two images with different poses based in [10]. We work under the assumption that our test environment has a planar surface. Thus, we use an homography to transform the image to an orthonormal view (top-down in our case) by means of four fixed reference points in the ground. This transformation works for the ground (if it is planar) but does not work for trees or other objects which are not in the same plane which will often experience several distortion (see bushes in rightmost images in fig. 6).



6: Axes alignment for two images. Images are rectified, transformed to a top-down view, and rotated to align their axes.



7: Normalized correlation between two images.

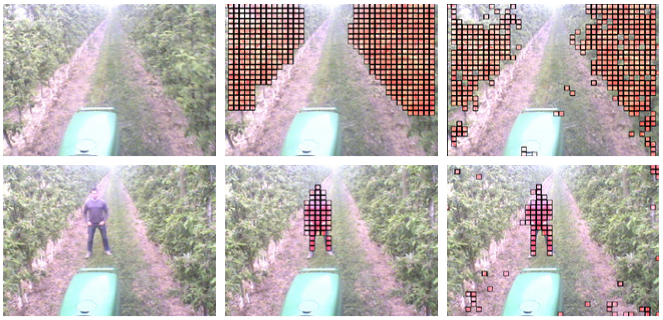
Once both images are transformed in these coordinates, the optical axes are parallel. We can use the yaw angle to rotate one of the images and align the axes of the two images as shown in fig. 6. Due to differences in the actual X, Y, Z coordinates where each image was taken, the resulting images may not align, yet. However, after these transformations, the camera axes are parallel and we can compensate for these differences in pose by a simple translation. We use normalized correlation to compute the relative translation of one image w.r.t. the other (see fig. 7 for details). Finally, we can relate each patch from the reference image to each one in the input image (see fig. 8).

In our experiments, images with differences in position smaller than 3m, and differences in yaw smaller to 10 degrees were successfully registered. Beyond these numbers, this method may fail to successfully recover the correct translation for every patch in one image to match the patches in the reference image. Hence, it is important to find images in the database which are very close in pose to the image that we are labeling.

VII. Experiments

In this section, we show that the use of contextual models improves the performance for obstacle detection tasks from images. We compare the different contextual models presented in this paper (CRF, KCRF, PK-CRF), a logistic regression classifier (LR) and the Import Vector Machine (IVM).

8: **Left:** Flow for every 16×16 pixels patch to match patches in the right image. **Right:** Reference image.



(a) Original images (b) CRF labelings (c) LR labelings

9: Red boxes denote detections. **Top row:** Orchard detection. **Bottom row:** People detection.

A. Data acquisition platform and features

In our experiments, we use data collected with a vehicle equipped with several sensors: a color camera, an Inertial Measurement Unit (IMU), a wheel encoder (for vehicle speed input), a scanning laser and a Global Positioning System (GPS).

Camera and laser sensors are registered w.r.t. each other and the vehicle. We use a Kalman filter to compute the local pose of the vehicle using the speed of the vehicle and the information collected by the IMU. Finally, the GPS is used to acquire the global position of the vehicle at the start time, thus we can reference the local position among different sequences of data collected at different times (we call them logs).

Every image is divided into a grid of patches of 16×16, and we extract feature information from each patch independently, as described in [3]. The features extracted contain the mean and standard deviation for the U,V components in the LUV color space and texture information for a total of 28 features. Every feature was scaled to have 0 mean and standard deviation of 1.

The ground truth contains binary labels (obstacle/not obstacle) for every patch in each image and it is automatically computed using the laser data. The 3D data extracted from the laser is very accurate and it is used to get a good estimation of the ground. Once the ground is estimated, the detection of obstacles becomes very simple (i.e. any 3D point above the ground more than 0.5 m is an obstacle). We project the object location to the image frame and use that information for automatically getting the labels for every image patch in the grid.

LR classifier is trained using the 28 features + a constant feature to account for the bias. Similarly, we use the same features for the CRF node potentials and a total of 57 edge features (28×2(classes) $+ 1$(bias)). The edge features were computed as the Euclidean distance of two adjacent node features. Therefore, the total number of variables of the linear CRF was 86, which were successfully learned using the algorithm described in section III-C.

B. Comparison of LR and CRF

We compare the performance of LR and CRF (trained using the same features as described in section VII-A) for obstacle detection in an agricultural application. One of the tasks is to be able to drive a vehicle (i.e. tractor) through orchard tree lanes. The classifier must properly segment and detect the orchard tree lanes. We acquired two sequences of images and we used one for training the classifiers and the second one for producing the video: `http://www.cs.cmu.edu/~cvalles/videos/orchard.avi`

As can be seen in the video and in fig. 9 (top row), CRF provides a cleaner segmentation of the orchard tree lanes. Furthermore, the number of false positives produced by the CRF classifier is much lower. LR produces false positives continuously, some of them just in front of the vehicle, which would make the vehicle to stop. However, CRF does not produce any false positive in front of the vehicle while properly segmenting the orchard tree lanes throughout the video sequence. CRF significantly outperformed LR for detection and segmentation of people in our experiments as it is shown bottom row of fig. 9.

C. Comparison of LR, CRF, IVM, KCRF and PKCRF

We collected 8 sequences of data in a mostly flat and grassy environment, driving at 2m/s for about 4 minutes, with several obstacles scattered around. We logged 2 images per second. Two sequences were used for training and the other 6 were used for testing. Our definition of obstacle in this environment is anything that is above the ground more than a certain distance (i.e. 0.5m). Thus, in our gathered data, the obstacles are bushes, cones, trees, vehicles, fences, etc (see fig. 10). The data was collected in two different days, and some of the obstacles were placed in different positions. We followed similar trajectory for every log we took, just allowing deviations from the original path smaller than 5m.

CRF and LR were trained as described in section VII-A. We used a Gaussian kernel for IVM, KCRF and PKCRF classifiers, and we experimentally found the optimal kernel width to be $\sigma = 3$ by plotting the histogram of distances mapped by the kernel as proposed in [5]. The IVM algorithm found 119 import vectors. K-CRF used an extra 57 variables for the edge potentials (same ones as in the linear case) and 1 for bias, totaling 177 variables. In the case of PK-CRF, one of the sequences was used as reference to extract features described in section V, and was used as database for the test

10: Some snapshots of the environment in which data was collected. Note that an obstacle can be anything that may be a hazard for the vehicle (bushes, cones, other vehicles, trees, etc.)

TPR	LR	IVM	CRF	KCRF	PKCRF
0.95	$61.6\% \pm 0.4$	$37.3\% \pm 0.7$	$37.0\% \pm 1.3$	$20.1\% \pm 2.8$	$13.5\% \pm 5.4$
0.92	$38.9\% \pm 0.5$	$16.6\% \pm 3.0$	$10.5\% \pm 6.1$	$4.3\% \pm 3.3$	$2.9\% \pm 2.4$
0.90	$24.3\% \pm 4.4$	$10.9\% \pm 2.2$	$4.1\% \pm 3.2$	$1.9\% \pm 1.6$	$1.6\% \pm 0.8$
0.88	$15.2\% \pm 3.5$	$7.2\% \pm 1.0$	$2.0\% \pm 1.6$	$1.2\% \pm 0.5$	$0.9\% \pm 0.3$
0.85	$9.5\% \pm 1.4$	$4.7\% \pm 0.5$	$1.0\% \pm 0.3$	$0.6\% \pm 0.2$	$0.4\% \pm 0.2$
0.80	$5.1\% \pm 0.6$	$2.8\% \pm 0.2$	$0.6\% \pm 0.1$	$0.3\% \pm 0.1$	$0.2\% \pm 0.1$
0.75	$3.3\% \pm 0.3$	$1.9\% \pm 0.1$	$0.4\% \pm 0.1$	$0.2\% \pm 0.1$	$0.1\% \pm 0.1$

I: False Positive Rate (FPR) for a given True Positive Rate (TPR) for each algorithm evaluated in this paper.

experiments.

In order to compare the performance of the different algorithms, we considered a false positive an alarm from the classifier in an area of a 3×3 image patches that does not contain an obstacle. In this application, a false positive may cause the vehicle to stop for no apparent reason, degrading the performance of the autonomous vehicle. However, a false negative may be fatal. Hence, it is very important to achieve high obstacle detection rates when working at low false positive rates.

Table I shows the false positive rate of the various classifiers at different performance points. In this case, differences among the classifiers become apparent, specially at low false positive rates. The false positive rate (FPR) for a fixed true positive rate (TPR) is shown in table II. Whereas neither LR nor IVM could be used in practice because of its large FPR at any performance point in the table, contextual methods give enough performance boost to be considered.

At low FPRs, contextual methods perform several times better than non-contextual ones. For instance, at a fixed FPR of $1/250$, LR and IVM achieve obstacle detection rates of 30% and 43%, respectively (see table II). Contextual methods (CRF, KCRF and PKCRF) give performances over 75%. PKCRF gives a performance just shy of 85% at the same FPR.

Also, note that PKCRF achieves an obstacle detection

FPR	LR	IVM	CRF	KCRF	PKCRF
1/1000	$4.5\% \pm 0.2$	$15.0\% \pm 1.3$	$57.1\% \pm 2.5$	$70.5\% \pm 3.3$	$75.3\% \pm 3.3$
1/750	$7.6\% \pm 0.50$	$21.2\% \pm 0.9$	$63.5\% \pm 2.8$	$73.2\% \pm 3.3$	$77.2\% \pm 3.3$
1/500	$12.7\% \pm 1.3$	$28.4\% \pm 1.5$	$68.5\% \pm 3.0$	$76.9\% \pm 3.0$	$80.0\% \pm 3.3$
1/250	$29.7\% \pm 1.5$	$43.3\% \pm 1.8$	$76.3\% \pm 2.7$	$82.1\% \pm 2.9$	$84.7\% \pm 3.0$
1/100	$50.2\% \pm 1.8$	$64.3\% \pm 2.2$	$84.7\% \pm 2.6$	$87.4\% \pm 2.8$	$88.2\% \pm 2.6$
1/75	$56.3\% \pm 2.0$	$70.2\% \pm 2.5$	$86.2\% \pm 2.6$	$88.7\% \pm 2.6$	$89.0\% \pm 2.7$
1/50	$66.46\% \pm 2.0$	$75.9\% \pm 2.6$	$88.1\% \pm 2.4$	$90.3\% \pm 2.5$	$91.0\% \pm 2.4$
1/25	$77.3\% \pm 2.3$	$83.6\% \pm 2.7$	$89.9\% \pm 2.4$	$91.8\% \pm 2.3$	$92.7\% \pm 2.1$
1/10	$85.5\% \pm 2.3$	$89.9\% \pm 2.3$	$91.9\% \pm 2.2$	$93.7\% \pm 1.9$	$94.4\% \pm 1.8$

II: True Positive Rate (TPR) for a given False Positive Rate (FPR) for each algorithm evaluated in this paper.

LR	IVM	CRF	KCRF	PKCRF
> 100 img/s	~ 20 img/s	~ 10 img/s	~ 3 img/s	~ 1 img/s

III: Number of processed images per second in a Intel Core 2 Duo class machine.

rate of 75% generating a false positive every 1000 positive-classified patches. IVM and LR produce 1 false positive every 50 and 25 positive-classified patches to get the same obstacle detection rate. At this performance point, PKCRF performs 20 and 40 times better, respectively.

We briefly evaluated the computational complexity of these methods. We ran these experiments in a Intel Core 2 Duo class machine running at 2.0 Ghz. Code was not optimized to make use of the two cores, though. Table III shows the number of images per second that every classifier was able to process. This timing does not take into account the time needed to pre-process the image and extract its features.

We included the full Receiver Operator Characteristic (ROC) curves in fig. 11 and in table IV, we show the Area Under the Curve(AUC) for each of the methods. Note that the differences among the various algorithms shown in this table are small, and AUC is not significative enough to establish conclusions. However, in practice, non-contextual algorithms do not offer enough performance to be used in practice in these experiments, due to its bad obstacle detection rates at low false alarm rates.

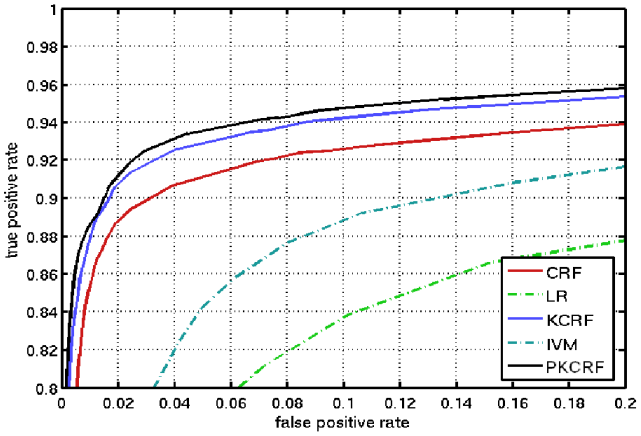
Finally, we generated a video for one of the sequences with the outputs of each classifier discussed in the paper: `http://www.cs.cmu.edu/~cvalles/videos/test-obs.avi` In this sequence, the vehicle was manually driven and there were obstacles at different locations. The video shows red patches for detections, and a blinking red box around the image of the classifier that produces a false positive that would make the vehicle to stop. A snapshot of the video is shown in fig. 12 where the outputs of the different classifiers evaluated in this paper are displayed.

VIII. Conclusion

In this work, we presented an obstacle detection algorithm for autonomous vehicles using a monocular color camera. We extended a CRF model to allow the use of non-linear kernels and prior data. In the experimental section, we showed that the use of lattice models for image labeling tasks helps to obtain globally more accurate segmentations than classifiers that make locally independent decisions. Furthermore, we showed that Gaussian kernels work better than linear kernels

LR	IVM	CRF	KCRF	PKCRF
$91.9\% \pm 1.4$	$94.9\% \pm 1.1$	$95.7\% \pm 1.2$	$97.0\% \pm 0.9$	$97.4\% \pm 0.8$

IV: Area Under the Curve (AUC) for each algorithm.



11: ROC curves for LR, IVM, CRF, KCRF and PKCRF.

in our obstacle detection experiments. Finally, prior data was introduced in the CRF model to produce better segmentations in "familiar" environments.

Even though the set of features we used for every image part was very limited (just color and texture features), the obtained results are promising towards building an obstacle detection system based only on a monocular color camera. Gaussian K-CRFs were the best performers when no prior data was available. However, PK-CRFs performed even better when prior data was available.

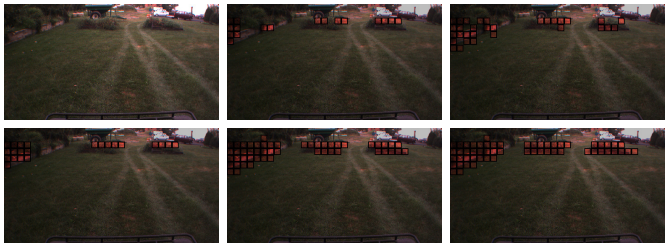
The algorithms proposed in this paper are not limited to monocular camera approaches. Future work includes the use of multiple camera solutions with different filters on them, the use of stereo features, experiments with different patch sizes (or non-uniform image divisions, such as super-pixels), the use of more node features, and extracting specific edge features for the CRF model. As one of the main issues when dealing with camera-only solutions is the exposure, High Dynamic Range (i.e. capture images at different exposures) could be used to address this problem.

In conclusion, CRFs provide a probabilistic framework with superior performance compared to classifiers that do not exploit context in image labeling applications. The model for CRFs is flexible enough to support different kernels, or the addition of a large variety of features. In our experiments, features that incorporate prior data helped to boost the obstacle detection performance, making PK-CRFs suitable for some robotic applications that use a monocular color camera.

Acknowledgments

We would like to acknowledge the valuable support of Cristian Dima and Carl Wellington in the development of the infrastructure used for the experiments, and for their helpful comments and advice.

This work was supported by John Deere under contract 476169.



12: **From left to right and top to bottom:** comparison of original image, LR, IVM, CRF, KCRF and PKCRF for one of the test images. Obstacles are marked with red boxes.

References

[1] C. Broyden. The convergence of a class of double-rank minimization algorithms. *Journal of the Institute of Mathematics and Its Applications*, 6:76–90, 1970.

[2] C. J. C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2:121–167, 1998.

[3] C. Dima, N. Vandapel, and M. Hebert. Sensor and classifier fusion for outdoor obstacle detection: an application of data fusion to autonomous off-road navigation. *The 32nd Applied Imagery Recognition Workshop (AIPR2003)*, October, 2003.

[4] R. Fletcher. A new approach to variable metric algorithms. *Computer Journal*, 13:317–322, 1970.

[5] D. Franois, V. Wertz, and M. Verleysen. About the locality of kernels in high-dimensional spaces. *ASMDA 2005, International Symposium on Applied Stochastic Models and Data Analysis*, pages 238–245, 2005.

[6] D. Goldfarb. A family of variable metric updates derived by variational means. *Mathematics of Computation*, 24:23–26, 1970.

[7] S. Kumar and M. Hebert. Discriminative fields for modeling spatial dependencies in natural images. *In Advances in Neural Information Processing Systems 16 (NIPS 2003)*, 2004.

[8] J. Lafferty, X. Zhu, and Y. Liu. Kernel conditional random fields: representation and clique selection. *In Proceedings of the Int. Conference on Machine Learning*, 2004.

[9] K. Murphy, Y. Weiss, and M. Jordan. Loopy belief propagation for approximate inference: An empirical study. *In proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pages 467–47, 1999.

[10] K. Primdahl, I. Katz, O. Feinstein, Y. Mok, H. Dahlkamp, D. Stavens, M. Montemerlo, and S. Thrun. Change detection from multiple camera images extended to non-stationary camera. *In Proceedings of Field and Service Robotics (FSR05)*, 2005.

[11] A. Quattoni, M. Collins, and T. Darrel. Conditional random fields for object recognition. *In Advances in Neural Information Processing Systems 17 (NIPS 2004)*, 2005.

[12] A. Saxena, S. Chung, and A. Ng. 3-d depth reconstruction from a single still image. *Int. J. Comput. Vision*, 76(1):53–69, 2008.

[13] D. Shanno. Conditioning of quasi-newton methods for function minimization. *Mathematics of Computation*, 24:647–656, 1970.

[14] A. Torralba, K. Murphy, and W. Freeman. Contextual models for object detection using boosted random fields. *In Advances in Neural Information Processing Systems 17 (NIPS 2004)*, 2005.

[15] S. Vishwanathan, N. N. Schraudolph, M. W. Schmidt, and K. Murphy. Accelerated training of conditional random fields with stochastic meta-descent. *In Proceedings 23rd Intl. Conf. Machine Learning (ICML)*, pages 969–976, 2006.

[16] Y. Wang and Q. Ji. A dynamic conditional random field model for object segmentation in image sequences. *In IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 1, 2005.

[17] J. Winn and J. Shotton. The layout consistent random field for recognizing and segmenting partially occluded objects. *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, 1:37–44, 2006.

[18] J. Zhu and T. Hastie. Kernel logistic regression and the import vector machine. *In Proceedings of Neural Information Processing Systems*, 2001.