Low-Power Parallel Algorithms for Single Image based Obstacle Avoidance in Aerial Robots

Ian Lenz,[1] Mevlana Gemici,[2] and Ashutosh Saxena.[1]
[1]Department of Computer Science, [2]Department of Electrical & Computer Engineering.
Cornell University, Ithaca, NY.
Email: `ianlenz@cs.cornell.edu, mcg74@cornell.edu, asaxena@cs.cornell.edu`

***Abstract*— For an aerial robot, perceiving and avoiding obstacles are necessary skills to function autonomously in a cluttered unknown environment. In this work, we use a single image captured from the onboard camera as input, produce obstacle classifications, and use them to select an evasive maneuver. We present a Markov Random Field based approach that models the obstacles as a function of visual features and non-local dependencies in neighboring regions of the image. We perform efficient inference using new low-power parallel neuromorphic hardware, where belief propagation updates are done using leaky integrate and fire neurons in parallel, while consuming less than 1 W of power. In outdoor robotic experiments, our algorithm was able to consistently produce clean, accurate obstacle maps which allowed our robot to avoid a wide variety of obstacles, including trees, poles and fences.**

I. INTRODUCTION

Perceiving obstacles is extremely important for an aerial robot in order to avoid collisions. Methods based on stereo vision [25], [9] are fundamentally limited by the finite baseline between the stereo pairs [22], and fail in textureless regions and in presence of specular reflections [4]. Active range-finding devices (e.g., [36], [20]) are either designed for indoor low-light environments (e.g., the Kinect [1]), or are too heavy for aerial applications. More importantly, they demand more onboard power, which is at a premium for aerial vehicles.

In this paper, we use a single monocular camera for obstacle perception. Recent works [31], [21], [29], [32], [30], [3] have shown that it is possible to obtain depth from a single monocular image. Multiple frames can also be used in combination to determine depth, but this approach does not work well on an aerial robot due to camera disturbances from robot motion and vibrations. Here, we present an algorithm that takes a single image as input and classifies each region in the image as obstacle or not. We will define an obstacle as an object which the robot could not safely pass through. This approach is very attractive for aerial robots because cameras are small and draw little power.

Our second key contribution is to formulate a Markov Random Field classification model and design fast inference algorithms for estimating obstacles in a new low-power parallel hardware that implements a leaky integrate and fire neuron architecture. One implementation of such hardware [19] consumes only 45 pico-Joules per spike (see Section III for an overview). The combination of using a camera (miniature cameras that consume extremely low power are available) and this hardware could allow miniature aerial robots to successfully fly amidst obstacles even in unknown environments. Inference in our MRF is solved using Belief Propagation (BP). While performing belief propagation in a traditional computer is expensive, our design allows the low-power hardware to do so natively. Each logical clock cycle performs a full parallel update of BP. We obtain good obstacle estimates once the BP network has converged (see Figure 1). We then use the estimated obstacle map to select a evasive manuever for avoiding the obstacle.

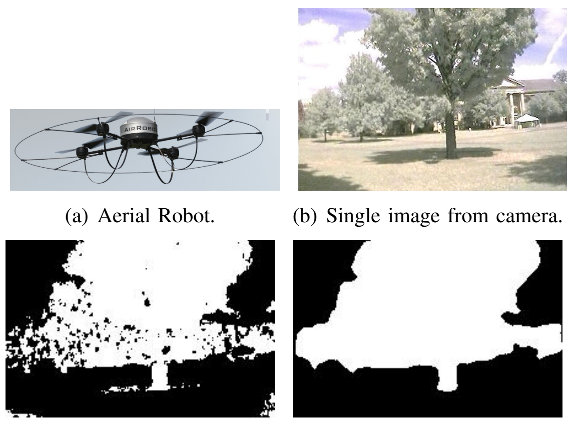
(a) Aerial Robot. (b) Single image from camera. (c) Initial inferred obstacle map. (d) Inferred obstacle map.

Fig. 1. We use our low-power parallel hardware for computing an obstacle map, given a single image from the camera onboard the aerial robot. We then use these results to select an evasive maneuver.

In detail, we represent the energy function of the MRF over the neuron architecture, and use spikes to propagate beliefs during inference. Our MRF uses the logistic function to model the local dependence of visual features to the obstacle label, where each spatial region of the image is represented using a different set of neurons—this allows parallel computation. Furthermore, we use different parameters for different spatial regions of the image [15] for improved performance. Finally, we induce sparsity in the parameters of the model to satisfy constraints on the number of parameters allowed by our hardware.

We performed extensive experiments in a variety of environments, containing obstacles such as trees, fences, and poles. In learning experiments, we obtained an average precision and recall of 81.9% and 93.6% respectively. In

53 outdoor robotic experiments, our algorithm was able to successfully perceive obstacles in every case, and avoid them in 51 cases. The two failures were due to communication delays and robot drift. Some of these experiments involved avoiding a series of obstacles of multiple types.

II. Related Work

In aerial robotics, most works which perform obstacle avoidance either make strong assumptions on precise knowledge of 3D location of obstacles [18], or use sensors that are not onboard, such as GPS (together with known obstacle map). Other work such as [8] focuses on mapping obstacles from overhead images. For a small aerial robot to fly autonomously in a real environment full of obstacles, these techniques do not directly apply.

Navigation by labeling obstacles in images has been used for several ground robots. For example, Ghosh and Mulligan [7] use a ground segmentation approach for navigation, while Nabbe and Hebert [26] use ground-vertical segmentation for extending the path planning horizon for ground robots. Work such as [12], [33] employs learning algorithms to determine terrain traversability for ground vehicles. Michels, Saxena and Ng [21] and Plagemann et. al. [28] attempt to determine range directly from monocular images. However, these works use only a local feature based classifier for navigating a ground vehicle. For an aerial robot such as ours, we are severely constrained by onboard power, and we present methods that allow even a complex inference method such as BP to be efficiently computed in low-powered hardware.

There are other works that consider single monocular image based obstacle avoidance for aerial robots. McGee et. al. [17] use sky segmentation for detecting obstacles, but apply only a local classifier. Soundararaj, Sujeeth and Saxena [34] and Courbon et. al. [6] use vision techniques to navigate aerial robots, but are limited to known indoor environments. Bills, Chen and Saxena [5] and Zingg et. al [37] perform similar work to unknown environments, but still handle only a few known types of indoor environment. On the other hand, we consider general outdoor environments, employing learning algorithms which allow our approach to be easily adapted to new obstacle types and integrate non-local information to enhance classification.

Vision algorithms have implemented in neural architectures or embedded systems, such as [13], [2], [11]. Other works [23], [10] used spiking neurons for basic obstacle detection and navigation. However, these approaches do not generalize to real-world outdoor cases.

III. Neuromorphic Hardware

Our goal is to develop algorithms for obstacle mapping that can be implemented in low-power parallel hardware. In particular, we use a neuromorphic hardware platform that comprises a network of linear-leak integrate and fire (LLIF) artificial neurons as in [19]. The LLIF neurons represent an extremely versatile high-level primitive which couples memory and processing. More importantly, this architecture uses extremely low power, as discussed below.

Each neuron integrates the weighted synaptic inputs from other neurons and fires if the integrated value exceeds a preset threshold. More formally, each neuron has some integer-valued internal potential Z and binary-valued spiking output S. For i^{th} neuron, S and Z update as:

$$\begin{aligned} Z_t^{i+} &= Z_t^i + \sum_{j \in \mathcal{N}(i)} w_{ij} Y_t^j - \lambda_i \\ Z_{t+1}^i &= \mathbf{1}\{Z_t^{i+} < \alpha_i\} Z_t^{i+} \\ S_{t+1}^i &= \mathbf{1}\{Z_t^{i+} \geq \alpha_i\} \end{aligned} \tag{1}$$

where $\mathcal{N}(i)$ are the neighbors of neuron i, $w_{i,j}$ indicates the synaptic weight from neuron j to neuron i, α indicates neuron threshold, and λ is a constant decay. $\mathbf{1}\{...\}$ is the indicator function, which takes the value one if its argument is true and zero otherwise.

Since these are spiking neurons, one major restriction is that the inputs and outputs be binary-valued. We address this by using representations where the spike count over a particular time window is proportional to the value being represented. Since each neuron can integrate over time, this is still a useful representation. The expected spike rate given the input x is simply a ramp function with limits, as follows:

$$g(x, \alpha) = \begin{cases} 0 & \text{if } x \leq 0 \\ x/\alpha & \text{if } 0 < x < \alpha_i \\ 1 & \text{if } x \geq \alpha_i \end{cases} \tag{2}$$

Cardinality constraints in hardware. To allow for a more compact design with lower power consumption, hardware such as that in [19] typically imposes a constraint on the cardinality of the weights w. That is to say, each neuron's weights may take at most k unique nonzero values.

Power consumption in hardware. In a well-designed hardware platform such as [19], power consumption will be proportional to the number of spikes and the density of connections. In particular, the hardware in [19] takes only 45 pico-Joules (pJ) per spike, and has very low quiescent power draw in their absence.

IV. Obstacle Classification

The primary goal of our approach is to produce an obstacle map of sufficient quality for obstacle avoidance. This is a challenging problem, as outdoor environments are perceptually complex, with variations in obstacle appearance, lighting, background appearance, and other factors. Our model will define obstacles as objects which project upwards from the ground and thus present a navigational challenge to the robot. We will use the labels it produces to select an evasive maneuver for the perceived obstacles.

Our classification model is a Markov Random Field (MRF) model (e.g., see [14]), where we use an Ising pairwise potential for modeling dependencies between neighboring

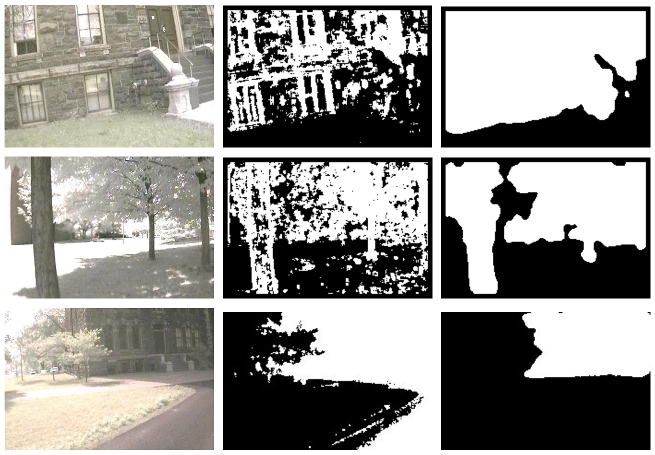

Fig. 2. Left: input images. Middle: initial classification. Right: classification with full MRF model with belief propagation. Initial classification results which would present problems for navigation, but are greatly improved by integrating non-local information using our MRF.

image regions. More formally, an MRF is an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where we represent each labeled location in the image as a vertex $\mathcal{V}$, and edges $\mathcal{E}$ connecting neighboring image locations. Let $Y^i \in \{-1, +1\}$ represent the binary labels indicating presence or absence of an obstacle at the i^{th} location in the image, and X^i represent the input visual features at that location.

We model the joint conditional likelihood of the labels given the features as:

$$P(Y|X,\theta) \propto \exp\left(-E(Y|X,\theta)\right) \tag{3}$$

where the $E(Y|X,\theta)$ is an energy function containing three terms:

$$E(Y|X,\theta) = -\sum_{i\in\mathcal{V}} Y^i A(X^i,\theta) - \sum_{(i,j)\in\mathcal{E}} w_{ij} Y^i Y^j + \beta||\theta||_1 \tag{4}$$

The first term uses a logistic model, with θ as its parameters, to model the dependence of the label on the local visual features. More formally, we model the association potential as $A(X,\theta) = 2*\sigma(X^T\theta)-1$, where $\sigma(x) = 1/(1+\exp(-x))$ is the sigmoid function. The second term prefers neighboring labels to be similar, with w indicating the relative importance of the two terms. Finally, β controls the strength of an L_1 regularization term on the local feature weights, which helps with the weight cardinality constraints in the hardware.

During learning, we will be given a set of labeled examples and our goal is to find the optimum value of the parameters. During inference, we are given a new image and our goal is to find the optimal value of Y using the LLIF neuron architecture.

A. Learning

We manually set parameters w_{ij}, and learn parameters θ by maximizing the pseudo-conditional log-likelihood. We are given M labeled ground-truth pairs as $\{(X_m, Y_m) : m = 1, \ldots, M\}$, and we learn θ^* as:

$$\theta^* = \arg\min_\theta \sum_{m=1}^{M} E(Y_m|X_m,\theta) \tag{5}$$

$$= \arg\min_\theta \sum_{m=1}^{M} \left(-\sum_{i\in\mathcal{V}} Y_m^i A(X_m^i,\theta)\right) + \beta||\theta||_1 \tag{6}$$

Here, Y_m^i indicates Y^i from the m^{th} training example, and X_m^i is similar for input features. This sub-problem is convex, and is equivalent to solving logistic regression with an additional L_1 penalty term. We vary the β parameter until the number of non-zero weights fits within what is allowed in the hardware.

B. Inference

The goal of inference is to find an optimal value for the label estimates Y, given features X and parameters θ:

$$Y^* = \arg\max_Y P(Y|X,\theta) \tag{7}$$

Our inference algorithm is based on loopy belief propagation [27], [24]. In order to derive the update rule for node i, we assume that the values of the other nodes are known and compute the messages as follows:

$$\psi(X^i) = \sigma(\theta^T X^i) \tag{8}$$

$$\mu_i(X^i) = \psi(X^i) \prod_{(i,j)\in\mathcal{E}} \mu_j(X^j)^{w_{ij}} \tag{9}$$

$$P(Y^i = 1|X) = \mu_i(X^i) \tag{10}$$

Our goal is to perform obstacle avoidance in new environments, and the product above may give zero probability of being an obstacle if any of terms is zero. This is not preferable, and we need to account for such cases. Following ideas of additive smoothing in statistics [35], we include an additive smoothing term in ψ and μ. This is a small constant factor ϵ added to each function. Denote the versions of these functions with additive smoothing as ψ_s and μ_s.

If we consider these update equations in log-space, they become sum of weighted terms, and thereby can be implemented in neuromorphic hardware (Eqn. 1). In such a case, the additive smoothing term becomes a constant lower bound on the log probability terms. Thus, in log-space, we have:

$$\log\psi_s(X^i) = \max(\log\epsilon, \log\sigma(\theta^T X^i)) \tag{11}$$

$$\log\mu_{s,i}(X^i) = \max\Big(\log\epsilon, \log\psi_s(X^i) + \sum_{(i,j)\in\mathcal{E}} w_{ij}\log\mu_{s,j}(X^j)\Big) \tag{12}$$

To implement this in hardware, we will use one neuron each to represent $\psi_s(X^i)$ and $\mu_{s,i}(X^i)$ for each node. Each ψ unit will take input in spike-rate from local features, weighted as θ. Each μ unit will take input from the corresponding ψ unit and neighboring μ units, weighted as w.

The log-sigmoid ψ_s term can be approximated as a linear function which saturates at 0. With the lower bound from the additive smoothing function, this becomes a scaled and shifted version of Eqn. 2. We will refer to the version of $g(x, \alpha)$ adjusted to fit the log-sigmoid function as $g_\psi(x)$. $\log \mu_{s,i}(X^i)$ is also well approximated by a thresholded linear function, and can thus also be modeled by $g(x, \alpha)$, as $g_\mu(x)$. In both cases, α is fixed to some value which gives the best fit. The equations for our system are then:

$$\log \psi_s(X^i) = g_\psi(\theta^T X^i) - \log Z_\psi \tag{13}$$

$$\log \mu_{s,i}(X^i) = g_\mu(w_{ii}\psi_s(X^i) + \sum_{(i,j)\in\mathcal{E}} w_{ij}\mu_{s,j}(X^j)) - \log Z_\mu \tag{14}$$

Where the Z's are two separate constants, necessary to include here to preserve exact equality, but unnecessary to implement in hardware since relative values are preserved. Except for the error introduced by approximating the log-sigmoid function with $g_\psi(x)$ and discretization, this is exactly Eqn. 12. To infer optimal values for Y, we can simply threshold the g_μ terms once the network has converged.

C. *Visual Features*

Our basic features are a standard set of texture filter responses, similar to those in [16]. They consist of oriented edge filters (in our case, Gabor filters) and center-surround filters (difference-of-Gaussian filters) as shown in Figure 3. We also include color information in the form of patch-averaged RGB values in our feature set. In addition to raw filter responses, we include the absolute value of these responses and the maximum and average of this absolute value over a local area.

We found it difficult to design features which fit the hardware constraints of [19], so the features given here do not. They do, however, rely only on simple local computations, and thus are amenable to efficient implementation in circuitry or parallel hardware. Only features given nonzero weights by some classifier would need to be implemented in this hardware, significantly reducing complexity.

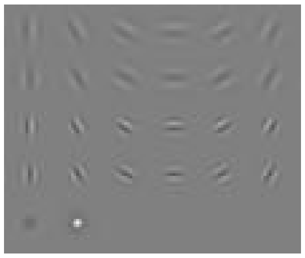

Fig. 3. Our filter set, which includes two scales of Gabor filters at six orientations and two scales of difference-of-Gaussian filters.

D. *Spatially-varying models and multiple models*

Often the statistics of the dependence of the obstacles on the visual features varies with their location in the image [15]. For example, while leaves on the trees are typically obstacles, they are not if they fall on the ground in the Fall season. Thus our parameters θ should be different for different rows in the image. In order to do so, we divide the image vertically into three equally sized areas. This is equivalent to considering a separate θ_i for each $i \in \mathcal{V}$, and tying weights within regions. By dividing the classification as such, each region can have a different feature set. Tying the parameters of these different models helps to avoid over-fitting. Since our neuron architecture is distributed for different spatial regions in the image, each neuron gets the appropriate synaptic weights depending on which region in the image it is representing.

To detect multiple visually different obstacle classes, we train a model for each class and use the combined output of these models. In order to exercise maximum caution, we combine outputs using the logical OR of the results, ie if any classifier classifies a region as obstacle, it is considered to be an obstacle for purposes of navigation.

Since our algorithms run natively in parallel hardware, neither of these changes require a change to the architecture or cause an increase in runtime.

E. *Tuning for Power Consumption*

One major strength of the hardware described in [19] is that its power consumption can be estimated a priori from simulation with high accuracy. This is because power consumption in this hardware is determined largely by two factors: a) connection density and b) spike count. While a) is fixed for a particular local connection pattern, b) can change drastically depending on the relative weight between local classifier estimates and neighboring labels, as well as neuron threshold and decay. In particular, faster convergence of the MRF model in terms of hardware timesteps can significantly reduce power consumption, as the model can be terminated earlier and thus generates fewer spikes. Therefore, in practice, we tune our models for a tradeoff between power and accuracy, aiming to produce results which produce fewer spikes while still yielding results which would be useful for obstacle avoidance.

In practice, we obtained good performance in terms of both accuracy and performance for the following parameter settings for each neuron:

- w_{ii}: set slightly below α (firing threshold).
- w_{ij}: set to a small local connectivity pattern (4- or 8-way), with uniform weights set to 10-20% of α.
- λ (decay): set to 5-10% of α.

V. Obstacle Avoidance Manuevers

The goal of our obstacle avoidance algorithm is to avoid obstacles in near to mid-range (e.g., about 2m to 10m). Our motion selection algorithm uses a library of paired motions and visual-space masks. If less than some threshold's worth of pixels within a masked area are labeled as obstacles, the corresponding motion is considered to be safe. The effects of sweeping this threshold are shown in Figure 6. A fixed preference order is used to select motions in cases where more than one is found to be safe.

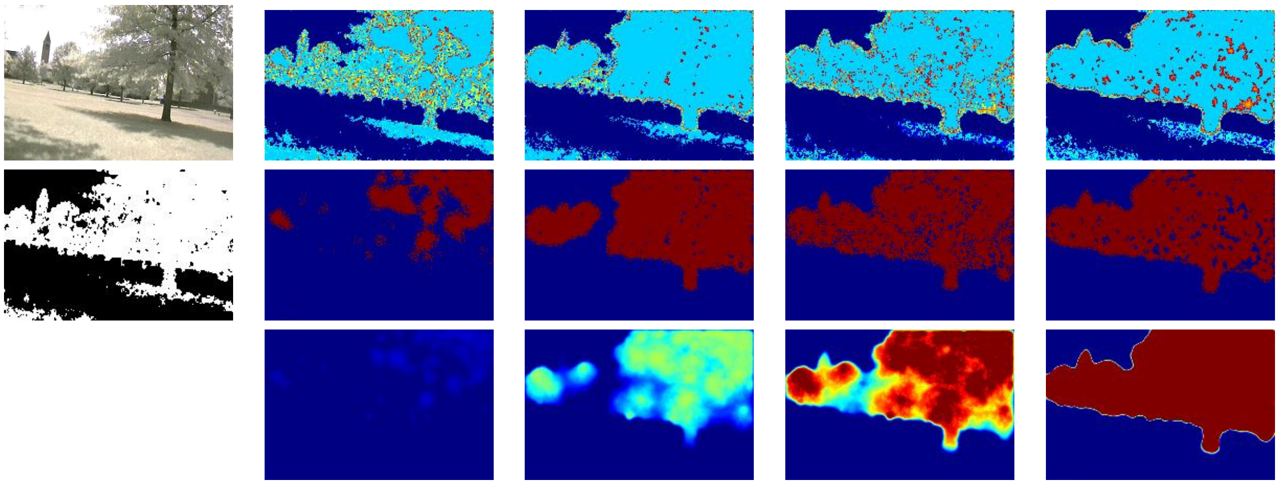
Fig. 4. Time series of belief propagation in action. Left: original image and baseline classification result. Top: internal node potential. Middle: spike locations. Bottom: spike counts. Time proceeds from left to right. Best viewed in color.

TABLE I

CLASSIFIER PRECISION AND RECALL, IN PERCENT, FOR FOUR OBSTACLE CLASSES, AND AVERAGE ACROSS CLASSES. RESULTS PRESENTED FOR BASELINE LOCAL CLASSIFIER AND MRF MODEL, WITH AND WITHOUT SPATIALLY VARYING MODEL, AND VARYING WEIGHT CARDINALITIES.

Algorithm	Card.	Trees		Buildings		Fences		Poles		Average	
		Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.	Prec.	Rec.
Chance	-	50.7	52.0	70.1	65.3	28.9	21.5	5.4	4.4	29.3	26.4
Local association term only, no spatial-varying model	110	80.0	81.5	83.6	91.0	61.5	92.6	56.6	94.7	71.8	88.2
	8	78.6	77.2	86.9	89.4	68.3	90.0	62.2	92.6	74.8	86.0
	5	77.4	75.4	90.3	73.2	67.6	90.1	47.9	95.6	70.8	86.2
	3	77.2	75.3	88.7	78.2	66.8	90.7	47.7	95.4	69.7	86.4
Local association term only, spatial-varying model	110	75.9	85.5	91.2	86.1	80.3	92.1	54.0	86.7	75.4	87.6
	8	74.1	84.0	91.7	85.3	78.2	93.0	59.8	86.6	75.9	87.2
	5	72.8	81.3	91.0	85.6	77.4	93.2	56.3	86.8	74.4	85.6
	3	70.5	77.9	90.7	79.9	66.6	93.4	57.0	86.8	71.2	84.5
Full MRF, no spatialy varying model	110	79.8	94.8	89.0	94.2	82.1	87.4	80.8	91.7	82.2	92.3
	8	76.3	91.9	84.8	95.0	89.2	85.3	82.1	90.7	83.5	91.1
	5	76.0	92.0	90.3	89.6	84.1	87.9	80.6	90.2	81.7	91.5
	3	71.5	88.1	87.1	90.0	90.2	89.2	81.8	89.9	82.2	90.7
Full MRF, spatialy varying model	110	81.9	94.0	87.7	96.3	95.0	94.0	63.1	90.1	81.9	93.6
	8	78.9	92.7	87.6	96.2	92.2	93.9	65.3	89.1	81.0	93.0
	5	79.3	92.0	87.2	95.6	91.1	93.6	65.7	89.6	80.8	92.8
	3	74.4	90.1	86.9	95.8	90.3	92.5	64.2	88.5	79.0	91.7

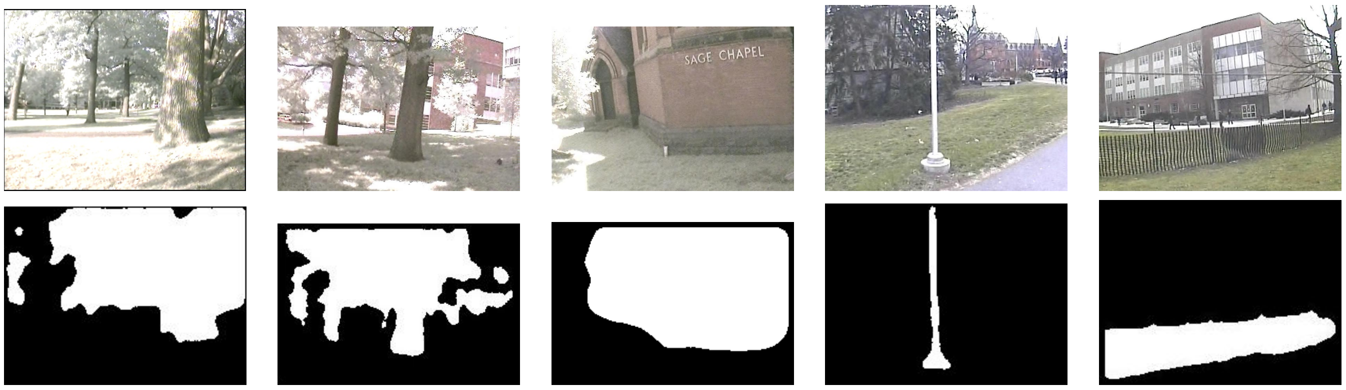

Fig. 5. Figure showing the results of our algorithm for a variety of obstacles. Top: Input image. Bottom: Inferred obstacle labels.

For vertical obstacles such as trees and poles, the motions, in order of preference, were diagonal-left, diagonal-right, and forwards. For horizontal obstacles such as fences, the motions were forwards or upwards.

VI. EXPERIMENTS AND RESULTS

A. *Offline Learning Experiments*

Dataset. Our dataset includes 120 images taken from various locations around the Cornell University campus using the onboard camera of our aerial robot. The environments include four types of obstacles: trees, buildings, light poles, and fences. The dataset includes 63 images of trees, 45 of buildings, 10 of poles and 10 of fences. We used 80% of the images for training and 20% for testing.

Belief Propagation. During inference, our BP system exhibits distinct phases of operation as seen in Figure 4. First, there is a warm-up phase where nodes with high values of classifier output build up energy. Eventually, some of these nodes' local potential exceeds their thresholds and they fire, spreading energy to neighboring nodes which may also fire in response. Spikes begin to propagate across the network, which finally reaches a steady state from which inferred classification labels are determined.

Results. Table I shows the performance of our algorithm. We present comparisons with different models. First, we consider a model with only the association term, i.e., equivalent to a local logistic classifier. Second, we also compare the effects of training several spatially varying models. We compare the effects of reducing the cardinality of local classifier weights as well. Similar baseline results were observed using SVM.

Our results show that the spiking neuron based BP system has proven very effective in producing cleaner, more usable results for obstacle avoidance, as compared to baseline results using the local logistic term only. Since our feature set contains edge and center-surround filter responses, local classifier estimates are generally stronger at obstacle edges. BP propagates these inwards, allowing the entire obstacle to be classified, causing increases in recall of as much as 17%. This propagation behavior can sometimes smooth edges of perceived obstacles slightly, as seen in many cases in Figure 5, causing a slight decrease in precision offset by gains elsewhere.

Most clear areas contain no strong local values and thus do not generate an initial spike, while true positive regions almost always do. This allows the system to avoid some false positive cases present in the baseline results. For obstacle avoidance purposes, recall is much more important than precision, and errors near an obstacle are less important than false positives in otherwise clear areas.

In order to evaluate the performance of our algorithm for obstacle avoidance purposes, we discretized the lower region of each test image into a 3x5 grid of cells, and considered a cell to be ground-truth occupied if it contained at least 10% ground-truth obstacles. We produced the curve shown in Figure 6 by sweeping thresholds for occupancy ratios. Our algorithm outperforms the baseline in all cases. The most significant improvement is for high values of recall, which are necessary for safe obstacle avoidance. Results presented are for trees, results for other classes were similar.

Weight Cardinality Limitations. Reducing the number of features available to the classifier decreases performance on average, producing particularly significant decreases for varied obstacles such as trees. However, with BP, the results are generally comparable for all weight cardinalities. This

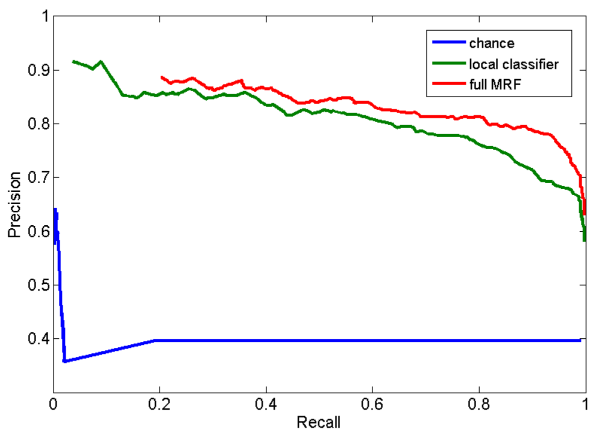

Fig. 6. Precision-recall curve for cell-based error metric, which demonstrates improvements over baseline results for navigation purposes.

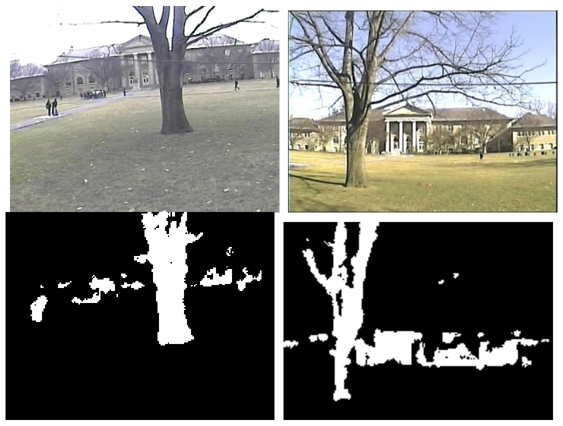

Fig. 7. Classification results for a tree trunk classifier using our algorithm, on the same tree under very different lighting conditions.

demonstrates that BP is able to resolve the errors produced by restricting weight cardinality, making it an ideal choice as an inference algorithm in neuromorphic hardware which imposes that constraint.

Spatially-varying models. Baseline results were improved by spatially-varying models in most cases, and full MRF results were improved in all cases except poles. This suggests that spatially varying models are useful in most cases, but might be foregone for obstacles with spatially consistent visual appearance such as poles.

Performance in different environments. Since our algorithm uses supervised learning to learn local classifier weights, it is easily adapted to new obstacle classes. The four classes presented here are very visually different, yet our model is able to detect obstacles effectively in each. Our model was also able to handle variations in lighting and obstacle appearance (such as leaves falling off the trees). Figure 7 shows an example of a tree trunk classifier using our algorithm performing effectively under a variation in lighting conditions.

B. *Robotic Experiments in Real Environments*

We performed obstacle avoidance experiments on our aerial robot platform, an AirRobot with a single onboard

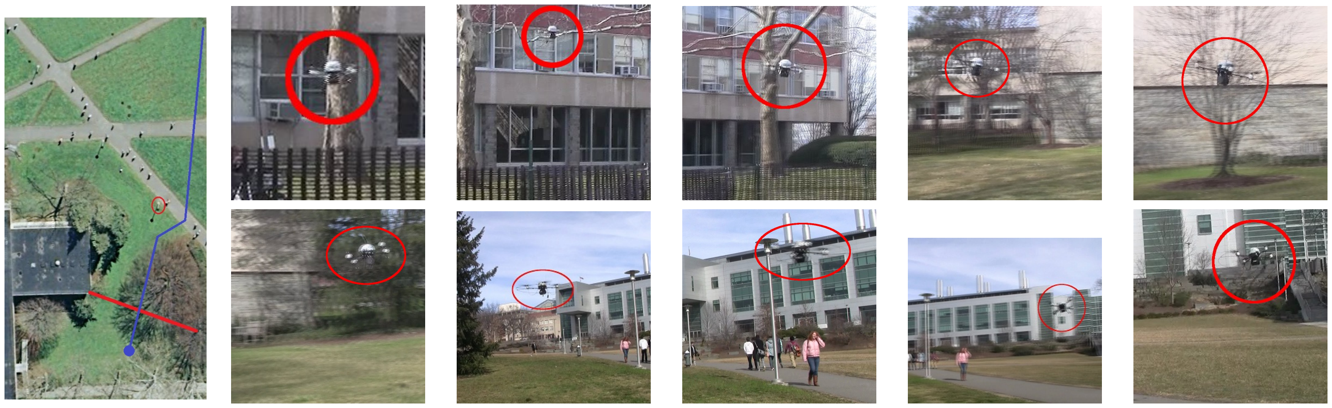

Fig. 8. Our aerial robot avoiding a fence and a pole in sequence. Left: Overhead map of area, red indicates obstacles avoided, blue robot path. Robot started at the blue dot, behind and below the level of the fence, moved upwards to a safe altitude, and proceeded over the fence and through the clear area. It then stopped at the pole, detected it as an obstacle, moved diagonally to the right to avoid it, and then forwards again through the following clear area.

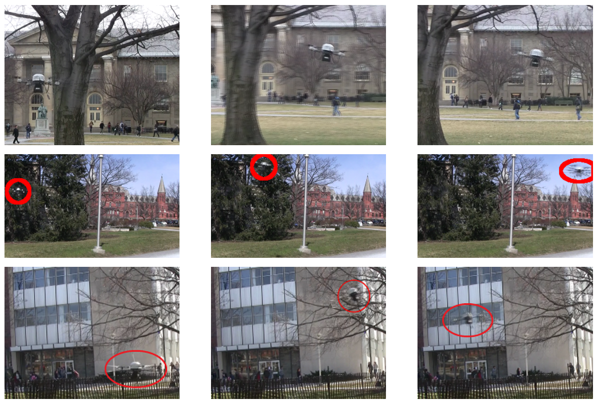

Fig. 9. AirRobot avoiding obstacles based on our classification results (robot circled in red in cases where it's difficult to see)

TABLE II
ROBOTIC EXPERIMENT RESULTS IN OUTDOOR ENVIRONMENTS. ERROR RATES PRESENTED FOR CLASSIFICATION, MOTION EXECUTION, AND OVERALL SUCCESSFUL AVOIDANCE

Type	Obstacles	Tests	Success Rate		
			Class.	Motion	Overall
Trees	3	20	100.0%	95.0%	95.0%
Fences	3	25	100.0%	100.0%	100.0%
Poles	2	8	100.0%	87.5%	87.5%
Total	8	53	100.0%	96.2%	96.2%

camera and an onboard IMU for stabilization.[1] Since the hardware described in [19] is still in production, processing was done using an offboard laptop computer running a MATLAB simulation of the hardware. Due to transmission latency and a lack of the necessary parallelism, this limited us to classifying a single image at a time, then taking a large step with the robot.

For each experiment, the robot was driven to a hovering position roughly 2-10 meters from the obstacle. Using the robot's IMU, we determined when it was in a stable, level pose, then captured an image from its onboard camera. This image was transmitted to the laptop, which performed the classification described in Section IV, using a classifier trained for the appropriate obstacle class. We then used the algorithm described in Section V to select an appropriate avoidance maneuver. The robot then executed a predetermined, fixed motion plan based on the maneuver chosen.

In Table II, we report three types of success-rates: "classification": when the obstacle classification was correct, "motion": when the computed obstacle avoidance maneuver was correct, and "overall": when the robot successfully executed the maneuver avoiding the obstacle.

In all the experiments in Table II, classification results from our model were of sufficient accuracy to allow the controller described in Section V to properly select a safe motion. Our model identified both foreground obstacles and clear areas consistently in these experiments.

As seen in Figure 9, our algorithm worked even in environments with visually cluttered backgrounds such as buildings and background trees. While these backgrounds did cause some false positives, as seen in some cases in Figure 5, these were only in the top region of the image, which was not considered by the controller. Since our algorithm works at visual range, it was able to perceive obstacles from long distances, allowing the robot to make large, predetermined motions to avoid them.

Some obstacles encountered were semi-transparent, such as the fence in the bottom row of Figure 9. Even though the visual appearance of the fence varied depending on viewing angle and background, and the fence exhibited specular reflections in some places, our algorithm was able to produce an extremely clean labeling of the fence as seen in Figure 5.

We also performed an experiment where the robot avoided multiple classes of obstacles in sequence. The classification models for fences and poles were run in parallel, and only motions determined to be safe by both were considered. The robot was re-oriented to its goal travel direction after each motion. The robot was able to travel roughly 25 meters while avoiding obstacles, as shown in Figure 8. This approach was

[1]Because of funding agency's restrictions, we are not allowed to disclose more detailed specifications of our system.

able to produce good results because the models were able to correctly report a lack of foreground obstacles when there were none, allowing only the classifiers for which foreground obstacles were present to dictate maneuver selection.

Video showing our robot avoiding obstacles using our algorithm is submitted as supplementary material, and is also available at: `http://mav.cs.cornell.edu`

VII. Conclusions

We presented a learning algorithm that takes as input a single image and outputs an obstacle map. Our algorithm uses a Markov Random Field to model the mapping from visual features to obstacles and the relations between neighboring regions in the image. We use parallel neuromorphic hardware for performing inference in the model. This hardware is well-suited for aerial robots because of its extremely low power requirements. Our MRF model also considers the cardinality constraints of the synaptic weights, and tries to minimize the power requirements for inference by minimizing the number of spikes. In upcoming hardware, our algorithms would allow several frames per second to be processed, while consuming less than 1 W of power.

We evaluated our algorithms in both learning experiments and robotic experiments. In learning experiments, our MRF model made significant improvements in classifier accuracy both quantitatively and qualitatively for the purpose of obstacle avoidance. In robotic experiments, our algorithm was able to correctly identify the locations of forgeround obstacles in all tests, allowing the robot to select an evasive maneuver to avoid them. Some of these tests involved multiple obstacles of different classes in sequence, demonstrating that inference results from multiple models can be effectively combined.

VIII. Acknowledgements

We would like to thank Dharmendra Modha, Shyamal Chandra, Thomas Zimmerman, Stefano Carpin, Steve Esser, Myron Flickner, Jerry Yeh, and Dale Cassidy for useful disussions, and Jasdeep Hundal and Brian Wojcik for their help with experiments. This work was partially supported by DARPA under grant #HR0011-09-C-0002 and by Alfred P. Sloan research fellowship to one of us (Saxena).

References

[1] A. Bachrach, S. Prentice, R. He, and N. Roy. Range - robust autonomous navigation in gps-denied environments. *Journal of Field Robotics*, 28(5):644–666, 2011.

[2] C. Bartolozzi, F. Rea, C. Clercq, M. Hofstätter, D. Fasnacht, G. Indiveri, and G. Metta. Embedded neuromorphic vision for humanoid robots. In *ECVW*, 2011.

[3] D. Batra and A. Saxena. Learning the right model: Efficient max-margin learning in laplacian crfs. In *CVPR*, 2012.

[4] D. Bhat and S. Nayar. Stereo in the presence of specular reflection. In *ICCV*, 1995.

[5] C. Bills, J. Chen, and A. Saxena. Autonomous mav flight in indoor environments using single image perspective cues. In *ICRA*, 2011.

[6] J. Courbon, Y. Mezouar, N. Guenard, and P. Martinet. Visual navigation of a quadrotor aerial vehicle. In *IROS*, 2009.

[7] S. Ghosh and J. Mulligan. A segmentation guided label propagation scheme for autonomous navigation. In *ICRA*, 2010.

[8] H. K. Heidarsson and G. S. Sukhatme. Obstacle detection from overhead imagery using self-supervised learning for autonomous surface vehicles. In *IROS*, 2011.

[9] S. Hrabar. 3d path planning and stereo-based obstacle avoidance for rotorcraft uavs. In *IROS*, 2008.

[10] S. B. i Badia, P. Pyk, and P. F. M. J. Verschure. A fly-locust based neuronal control system applied to an unmanned aerial vehicle: the invertebrate neuronal principles for course stabilization, altitude control and collision avoidance. *IJRR*, 26(7):759–772, 2007.

[11] T. Jochem, D. Pomerleau, and C. Thorpe . Vision-based neural network road and intersection detection and traversal. In *IROS*, 1995.

[12] D. Kim, J. Sun, S. Min, O. James, M. Rehg, and A. F. Bobick. Traversability classification using unsupervised on-line visual learning for outdoor robot navigation. In *ICRA*, 2006.

[13] A. Konno, R. Uchikura, T. Ishihara, T. Tsujita, T. Sugimura, J. Deguchi, M. Koyanagi, and M. Uchiyama. Development of a high speed vision system for mobile robots. In *IROS*, 2006.

[14] S. Kumar and M. Hebert. Discriminative random fields: A discriminative framework for contextual interaction in classification. In *ICCV*, 2003.

[15] C. Li, A. Saxena, and T. Chen. θ-mrf: Capturing spatial and semantic structure in the parameters for scene understanding. In *NIPS*, 2011.

[16] J. Malik, S. Belongie, J. Shi, and T. Leung. Textons, contours and regions: Cue integration in image segmentation. In *ICCV*, 1999.

[17] T. McGee, R. Sengupta, and K. Hedrick. Obstacle detection for small autonomous aircraft using sky segmentation. In *ICRA*, 2005.

[18] D. Mellinger, N. Michael, and V. Kumar. Trajectory generation and control for precise aggressive maneuvers with quadrotors. In *ISER*, Dec 2010.

[19] P. Merolla, J. Arthur, F. Akopyan, N. Imam, R. Manohar, and D. Modha. A digital neurosynaptic core using embedded crossbar memory with 45pj per spike in 45nm. In *CICC*, 2011.

[20] T. Merz and F. Kendoul. Beyond visual range obstacle avoidance and infrastructure inspection by an autonomous helicopter. In *IROS*, 2011.

[21] J. Michels, A. Saxena, and A. Y. Ng. High speed obstacle avoidance using monocular vision and reinforcement learning. In *ICML*, 2005.

[22] R. Moore, S. Thurrowgood, D. Bland, D. Soccol, and M. Srinivasan. A stereo vision system for uav guidance. In *IROS*, 2009.

[23] R. Mudra, R. Hahnloser, and R. J. Douglas. Neuromorphic active vision used in simple navigation behavior for a robot. In *Proc. 7th Int. Conf. On Microelectronics for Neural Networks*, pages 7–9, 1999.

[24] K. P. Murphy, Y. Weiss, and M. I. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *UAI*, 1999.

[25] I. Na, S. H. Han, and H. Jeong. Stereo-based road obstacle detection and tracking. In *ICACT*, 2011.

[26] B. Nabbe and M. Hebert. Extending the path-planning horizon. volume 26, pages 997–1024, 2007.

[27] J. Pearl. Fusion, propagation, and structuring in belief networks. *Artif. Intell.*, 29(3):241–288, 1986.

[28] C. Plagemann, F. Endres, J. M. Hess, C. Stachniss, and W. Burgard. Monocular range sensing: A non-parametric learning approach. In *ICRA*, 2008.

[29] A. Saxena, S. Chung, and A. Ng. Learning depth from single monocular images. In *NIPS*, 2005.

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

[31] A. Saxena, M. Sun, and A. Ng. Make3d: Learning 3d scene structure from a single still image. *PAMI*, 2009.

[32] A. Saxena, M. Sun, and A. Y. Ng. Make3d: Depth perception from a single still image. In *AAAI*, 2008.

[33] B. Sofman, E. L. Ratliff, J. A. D. Bagnell, J. Cole, N. Vandapel, and A. T. Stentz. Improving robot navigation through self-supervised online learning. *Journal of Field Robotics*, 23(12), 2006.

[34] S. P. Soundararaj, A. K. Sujeeth, and A. Saxena. Autonomous indoor helicopter flight using a single onboard camera. In *IROS*, 2009.

[35] V. Vapnik. *Statistical Learning Theory*. John Wiley and Sons, 1998.

[36] K. M. Wurm, R. Kümmerle, C. Stachniss, and W. Burgard. Improving robot navigation in structured outdoor environments by identifying vegetation from laser data. In *IROS*, 2009.

[37] S. Zingg, D. Scaramuzza, S. Weiss, and R. Siegwart. Mav navigation through indoor corridors using optical flow. In *ICRA*, 2010.